Fábio Vieira Teixeira

Infraestrutura de Rede de Sensores Sem Fio para Ambientes Assistivos

Campinas
2013

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Fábio Vieira Teixeira

Infraestrutura de Rede de Sensores Sem Fio para Ambientes Assistivos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação

Orientador: Prof. Dr. Eleri Cardozo

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Fábio Vieira Teixeira, e orientada pelo Prof. Dr. Eleri Cardozo

_____

Campinas
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

T235i

Teixeira, Fábio Vieira, 1987-
Infraestrutura de rede de sensores sem fio para ambientes assistivos / Fábio Vieira Teixeira. – Campinas, SP : [s.n.], 2013.

Orientador: Eleri Cardozo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Rede de sensores sem fio. 2. Robôs móveis. I. Cardozo, Eleri,1954-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Wireless sensor network infrastructure for assistive environments
Palavras-chave em inglês:
Wireless sensor network
Mobile robots
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Eleri Cardozo [Orientador]
Jó Ueyama
Paulo Cardieri
Data de defesa: 19-12-2013
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Fabio Vieira Teixeira

Data da Defesa: 19 de dezembro de 2013

Título da Tese: "Infraestrutura de Rede de Sensores Sem Fio para Ambientes Assistivos"

Prof. Dr. Eleri Cardozo (Presidente): _____

Prof. Dr. Jó Ueyama: _____

Prof. Dr. Paulo Cardieri: _____

Resumo

Na última década, as redes de sensores sem fio foram alvo de várias pesquisa em diversas áreas, entre elas a Robótica Móvel e Ambientes Inteligentes. Com o uso dessa tecnologia é possível estender as capacidades de sensoriamento de robôs móveis para o ambiente em que se encontram, além de aumentar a abrangência da conectividade sem fio. Devido à grande exploração da área e o grande avanço das tecnologias CMOS, novos módulos de sensoriamento, como câmeras, puderam ser desenvolvidos e embarcados em transceptores de baixo custo. A interconexão dessas fontes compostas de múltiplas mídias deu origem a uma nova tecnologia, as Redes de Sensores Multimídia Sem Fio. Essas redes possibilitam o desenvolvimento de uma grande quantidade de aplicações que antes não eram possíveis apenas com o uso de sensores escalares, por exemplo, em áreas relacionadas ao controle de tráfego, vigilância, automação residencial e cuidados com a saúde. Esta dissertação explora o uso desta nova tecnologia para o desenvolvimento de uma infraestrutura capaz de prover serviços a robôs móveis semi-autônomos em ambientes instrumentados para acessibilidade, a fim de auxiliá-los no cumprimento de seus objetivos.

Palavras-chave: Robótica Móvel. Ambientes Inteligentes. Redes de Sensores Multimídia Sem Fio. Assistividade.

Abstract

In the last decade, Wireless Sensor Networks have been the subject of research in many areas, including Mobile Robotics and Smart Environments. Such technology enables the extension of the mobile robots' sensing capabilities towards the environment where they are operating, as well as the increasing of the wireless connectivity coverage. Due to the large exploration of the area and the steady advance of CMOS technologies, new sensing modules such as cameras have being developed and embedded into low cost transceivers. The interconnection of multiple media sources raised a new technology known as Wireless Multimedia Sensor Networks. These networks allow the development of a wide range of applications that were not possible before using only scalar sensors, especially in areas related to traffic control, surveillance, home automation, and health care. This dissertation explores this new technology for the development of an infrastructure that provides services to semi-autonomous mobile robots in environments instrumented for accessibility, in order to assist the accomplishment of the robot's goals.

Key-words: Mobile Robotics. Smart Environments. Wireless Multimedia Sensor Networks. Assistivity.

Sumário

Introdução **1**

1 Redes de Sensores Sem Fio **5**
1.1 Definição de RSSF 5
1.2 Características Básicas das RSSF 8
1.3 Plataformas de *Hardware* 10
1.4 Sistemas Operacionais para RSSF 12
1.5 Pilha de Protocolos para RSSF 15
1.5.1 Camada Física 16
1.5.2 Camada de Controle de Acesso ao Meio 19
1.5.3 Camada de Rede 22
1.5.4 Camada de Transporte e Camada de Aplicação 27
1.6 RSMSF e Aplicações para Assistividade 29

2 Infraestrutura de Rede **36**
2.1 Plataforma 36
2.2 Sistema Operacional 38
2.3 Comunicação 40
2.4 Processamento de Imagem 42

3 Serviços Oferecidos pela Rede **46**
3.1 Serviços Desenvolvidos 46
3.1.1 Serviço de Navegação 47
3.1.2 Serviço de Localização 52

4 Experimentos e Resultados **57**
4.1 Navegação 57
4.2 Localização 62

Conclusões e Perspectivas **69**

Bibliografia **72**

Aos meus pais, Joaquim e Lourdes, e aos meus irmãos, Bruno e Bernardo.

Agradecimentos

Agradeço,

À minha família, por todo o amor, confiança e palavras de incentivo que me ergueram nos momentos mais difíceis, por estarem sempre presentes nas minhas vitórias e derrotas, e por sempre me ajudarem a escolher o melhor caminho a seguir.

Ao meu orientador, Prof. Eleri Cardozo, pela oportunidade de estudar na melhor instituição de ensino do país, pela paciência e sabedoria, e por todo seu conhecimento compartilhado, que contribuiram tanto para o meu crescimento profissional como pessoal.

À minha namorada Juliana, por todo amor, carinho e apoio em todos os momentos, que me ergueram e me incentivaram a seguir em frente.

Aos meus amigos de trabalho, Ricardo, Leonardo, Diego, Eric, Fernando, Guilherme e Lúcio, que sempre me ajudaram de todas as formas possíveis durante o desenvolvimento deste trabalho, contribuindo sempre com sugestões e experiências, além dos momentos de descontração proporcionados.

À professora Eliane Guimarães, pelo incentivo e apoio.

Ao professor e amigo Elionai Sobrinho, pelo seu amor e dedicação à arte de ensinar.

Aos professores Max Costa e Dalton Arantes, por acreditarem no sonho de seus alunos.

Aos meus amigos de longa data, Danilo Magalhães, Adriana Reis, Tamires Arias, Lucas Goes, Felipe Bergh e Victor Oliveira que sempre estiveram comigo de alguma forma.

À todas as pessoas que me acompanharam durante esse anos de estudo contribuindo para o meu progresso, em especial aos amigos Carlos Eduardo, Diogo Lima, Geraldo Vieira, Carol Leme, Gina Quelal e Mitchell Calderon.

À todos os amigos da Samsung Electronics que me incentivaram e me apoiaram, em especial ao amigo Moisés Danziger.

À CAPES, pelo apoio financeiro.

O correr da vida embrulha tudo, a vida é assim: esquenta e esfria, aperta e daí afrouxa, sossega e depois desinquieta. O que ela quer da gente é coragem.

Guimarães Rosa

Lista de Figuras

1.1 Arquitetura típica de uma rede de sensores sem fio. 6
1.2 Arquitetura básica de um nó sensor. 7
1.3 Modelo baseado em multitarefas. 13
1.4 Modelo baseado em eventos (Karl & Willig 2005). 13
1.5 Diagrama de blocos de uma comunicação RF (Akyildiz & Vuran 2010). 18
1.6 Topologias IEEE 802.15.4: (a) Estrela, (b) Malha, (c) Árvore. 22
1.7 Arquitetura típica para uma RSMSF. 30

2.1 RISC x CISC. 37
2.2 *Kit* da Memsic: (a) Imote2, (b) IIB2400, (c) ITS400, (d) IMB2400. 38
2.3 Instalação do sistema Linux na memória do Imote2. 39
2.4 Cenário para instalação do Linux. 40
2.5 SSH utilizando Ethernet-*over*-USB. 43
2.6 Imagens obtidas a partir da câmera do Imote2: (a) Imagem original, (b) Tons de cinza, (c) Negativo, (d) Detecção de bordas. 44

3.1 Cenário geral da aplicação de auxílio à navegação. 48
3.2 Cálculo do custo do caminho pelo algoritmo D*. 49
3.3 Subtração de plano de fundo: (a) Imagem base, (b) Imagem capturada, (c) Imagem resultante após subtração de pixels, (d) Imagem binarizada. 51
3.4 Robô Seekur Jr. com haste anexada. 52
3.5 Resultados da extração e detecção da marca: (a) Marca original. (b) Imagem da marca em sistema de cores BGR, (c) Marca detectada, (d) Marca extraída. . . . 53
3.6 Função potencial que representa a relação entre o diâmetro e a distância métrica da esfera. 54
3.7 Imagens obtidas a partir duas posições distintas: (a) Esfera posicionada no centro do ângulo de abertura da câmera, (b) Esfera posicionada no limite do ângulo de abertura da câmera. 54
3.8 Representação dos eixos globais do ambiente na imagem da marca. 55

4.1 Resultado do monitoramento de um corredor para diferentes ocupações. 58
4.2 Resultados da simulação de recálculo de rota: (a) Rota inicial gerada, (b) Recalculo de rota, (c) Comparação das rotas calculadas , (d) Rota final gerada 59

4.3 Mapa do ambiente de teste utilizado. 60
4.4 Rota inicial calculada através do algoritmo D*. 61
4.5 Recalculo de rota efetuado devido ao alto índice de movimentos no corredor intermediário. 61
4.6 Recalculo de rota efetuado devido ao alto índice de movimentos no corredor superior. 62
4.7 Pontos definidos com base nas medidas obtidas a partir da planta baixa do ambiente. 63
4.8 Estimativas da localização do robô sob a perspectiva do nó A. 64
4.9 Estimativas da localização do robô sob a perspectiva do nó B: (a) Estimativa da posição dos nós 8 e 10, (b) Estimativa da posição do nó 12. 66
4.10 Estimativas da localização do robô sob a perspectiva do nó C: (a) Estimativa da posição dos nós 8 e 10, (b) Estimativa da posição do nó 12. 67

Lista de Tabelas

1.1 Plataformas de Hardware. 11
1.2 Simuladores para RSSF. 28

2.1 Operações implementadas do AODV. 42

4.1 Resultados do erro Euclidiano. 65
4.2 Resultados do erro nos eixos. 65
4.3 Resultados do erro Euclidiano. 66
4.4 Resultados do erro nos eixos. 67
4.5 Resultados do erro Euclidiano. 68
4.6 Resultados do erro nos eixos. 68

Lista de Acrônimos e Notação

AODV	Ad Hoc On Demand Distance Vector Protocol (Protocolo Ad Hoc de Vetor de Distância Sob Demanda)
CISC	Complex Instructions Set Computer (Computador de Conjunto de Instruções Complexas)
CMOS	Complementary Metal-Oxide Semiconductor (Semicondutor Complementar Óxido-Metal)
CPU	Central Process Unit (Unidade Central de Processamento)
CTP	Collection Tree Protocol (Protocolo de Árvore de Coleções)
CSMA	Carrier Sense Multiple Access (Múltiplo Acesso por Sentido de Portadora)
DYMO	Dynamic MANET On Demand Protocol (Protocolo Sob Demanda para MANET Dinâmica)
DSDV	Destination-Sequenced Distance Vector Protocol (Protocolo de Vetor de Distância Sequenciado pelo Destino)
DSR	Dynamic Source Routing Protocol (Protocolo de Roteamento de Originador Dinâmico)
DSSS	Direct Sequency Spread Spectrum (Espalhamento de Espectro de Sequência Direta)
ETT	Expected Transmition Time (Tempo de Transmissão Esperado)
FFD	Full Function Device (Dispositivo de Função Completa)
FHSS	Frequency Hopping Spread Spectrum (Espalhamento de Espectro de Salto em Frequência)
GCC	GNU Compiler Collection (Coleção de Compilador GNU)
GUI	Graphical User Interface (Interface Gráfica de Usuário)
IEEE	Institute of Electrical and Electronic Engineers (Instituto dos Engenheiros Eletricistas e Eletrônicos)
IP	Internet Protocol (Protocolo de Internet)
MAC	Media Access Control (Controle de Acesso ao Meio)
MANET	Mobile Ad Hoc Networks (Redes Ad Hoc Móveis)
PC	Personal Computer (Computador Pessoal)
RAM	Random Access Memory (Memória de Acesso Aleatório)
RF	Radiofrequência
RFC	Request for Comments (Requisição por Comentários)
RFD	Reduced Function Device (Dispositivo de Função Reduzida)
RISC	Reduced Instructions Set Computer (Computador de Conjunto de Instruções

	Reduzidas)
ROM	Read Only Memory (Memória somente de Leitura)
RSMSF	Rede de Sensores Multimídia Sem Fio
RSSF	Rede de Sensores Sem Fio
SDRAM	Synchronous Dynamic Random Access Memory (Memória de Acesso Randômico Dinâmico Síncrono)
SMP	Sensor Management Protocol (Protocolo de Gerenciamento de Sensor)
SQDDP	Sensor Query and Data Dissemination Protocol (Protocolo de Disseminação de Dados e Consulta de Sensor)
SSH	Secure Shell (Shell Seguro)
TADAP	Task Assignment and Data Advertisement Protocol (Protocolo de Divulgação de Dados e Atribuição de Tareka)
TCP	Transport Control Protocol (Protocolo de Controle de Transporte)
TDMA	Time Division Multiple Access (Múltiplo Acesso por Divisão no Tempo)
TORA	Temporally Ordered Routing Algorithm (Algorimo de Roteamento de Ordenação Temporária)
UART	Universal Asynchronous Receiver/Transmitter (Receptor/Transmissor Assíncrono Universal)
USB	Universal Serial Bus (Barramento Serial Universal)
XML	Extensible Markup Language (Linguagem de Marcação Extensível)

Lista de Símbolos

α	Representa o ângulo de entrada do controlador *fuzzy*
ϕ	Representa o ângulo dado pela odometria do robô
Θ	Representa o ângulo da câmera no eixo horizontal
sin	Indica a função seno
cos	Indica a função cosseno

Introdução

Na atualidade, o processamento de dados se dá através de uma grande variedade de dispositivos computacionais de propósitos gerais, que vão desde *mainframes* até *smartphones* e computadores portáteis. Em aplicações como as de escritório, por exemplo, o processamento dos dados é efetuado de acordo com instruções recebidas do usuário do sistema na medida em que ele insere conteúdo em um arquivo, conteúdo este que na maioria das vezes está de alguma forma relacionado a processos físicos. Outra abordagem, ainda emergente, pode ser evidenciada quando o ambiente é o principal foco da aplicação. Nesse caso, a computação é usada para lidar com dados gerados a partir de processos físicos do ambiente, ao invés de processos lógicos criados por um usuário. Para que tal abordagem seja viabilizada, o mecanismo de computação precisa fazer parte de um sistema de controle, ou seja, deve estar embarcado em um controlador equipado com uma interface capaz de interagir com o ambiente monitorado. Um sistema de controle pode até mesmo ser programado para atuar sem intervenção humana.

Os sistemas embarcados em geral são bem difundidos e vêm sendo utilizados há várias décadas na engenharia. Estima-se que 98% dos sistemas de computação possuem uma versão embarcada (Karl & Willig 2005). O impacto desses sistemas na vida cotidiana está crescendo em um ritmo bastante acelerado e o progresso tecnológico vem proporcionando uma série de melhorias, o que possivelmente irão torná-los ainda mais comuns no nosso cotidiano. Sistemas como estes podem estar inseridos em ambientes diversificados, tais como locais de trabalho, residências e espaços públicos, provendo certo nível de inteligência ao ambiente. Nesse tipo de cenário, um conjunto de vários componentes podem extrair informações de diferentes fontes,

disponibilizá-las aos usuários e ainda, auxiliar na tomada de decisão e no controle de processos físicos.

O emprego de sistemas embarcados aborda três paradigmas de interação bem conhecidos na literatura: pessoa-pessoa, pessoa-máquina e máquina-máquina (Srivastava, Muntz & Potkonjak 2001). Para que esses paradigmas sejam atendidos, um aspecto crucial é necessário, a comunicação, dado que todas as fontes de informações têm que ter seus dados transferidos para um lugar apropriado, por exemplo, um atuador, um usuário, ou mesmo um sistema robótico. Em algumas situações, essas redes de dispositivos são facilmente implementadas de forma cabeada, porém, na maioria das vezes esse tipo de instalação apresenta um grande entrave ao sucesso do sistema, dado que acrescenta um alto custo financeiro devido a necessidade de compra e instalação do cabeamento (Rabaey, Ammer, Patel & Roundy 2000). A grande quantidade de componentes interconectados que geralmente são utilizados para monitorar um ambiente, também torna a utilização de cabeamento um grande problema para a manutenção, além, é claro, de não permitir mobilidade. Dessa forma, pode-se dizer que a comunicação sem fio é um importante requisito dos sistemas atuais. Nesse contexto, uma nova classe de redes cresceu nos últimos anos: as Redes de Sensores Sem Fio (RSSF). As RSSF têm recebido bastante atenção da comunidade, apresentando um crescimento anual de aproximadamente 42% nos últimos anos (Watteyne 2008). Uma RSSF é a interconexão de dispositivos miniaturizados, autônomos, capazes de realizar sensoriamento, processamento e transmissão de dados através de um enlace sem fio (Dargie & Poellabauer 2010). Com o uso dessa tecnologia é possível que cada nó de uma rede monitore sua região próxima e se comunique com outros nós de forma colaborativa para criar uma representação dos estados de um ambiente. Com essas características as RSSF podem prover inteligência ao ambiente, que passa a ser capaz de interagir com outras entidades presentes no sistema em que estão empregadas, como robôs, por exemplo, a fim de fornecer informações referentes ao sistema monitorado. Quando um ambiente é capaz de monitorar seu próprio estado e usar os dados obtidos para gerar informações capazes de auxiliar pessoas com algum tipo de restrição, como de locomoção, por exemplo, na execução de sua tarefa, este ambiente é considerado assistivo (Vangelis Metsis & Makedon 2008). Essa tecnologia

vem crescendo rapidamente e traz benefícios para diversas áreas, como a robótica móvel, por exemplo. Os robôs, dotados de capacidade de locomoção, se tornaram ainda mais presentes na indústria, hospitais e em residências, sendo considerados uma boa alternativa na execução de determinadas tarefas, como assistência de pessoas deficientes e auxílio na execução de atividades domésticas. Com a integração dos robôs móveis nos ambientes de convívio humano, uma grande variedade de métodos passou a ser estudados com o objetivo de lidar com a dinamicidade e com as várias restrições referentes ao tráfego que locais dessa natureza impõem. Diferentemente dos ambientes estáticos onde as posições dos obstáculos são bem conhecidas, nesse tipo de ambiente, novos e diferentes obstáculos podem surgir em qualquer instante causando obstruções e inviabilizando rotas, o que exige dos robôs a capacidade de se adaptar a essas mudanças repentinas rapidamente de modo que possa manter sua navegação de forma eficiente e natural até o cumprimento de seu objetivo (Svenstrup, Bak & Andersen 2010). RSSF tem sido considerada uma boa alternativa para auxiliar na solução de tais problemas através do monitoramento contínuo do ambiente.

Recentemente, o grande avanço das tecnologias CMOS permitiu o desenvolvimento de módulos multimídia de baixo custo, condizentes com os requisitos das RSSF. Com esses novos recursos disponíveis, uma ampla variedade de novas aplicações se tornaram viáveis através da integração de som, imagem e vídeo. Neste novo cenário surge o conceito de Redes de Sensores Multimídia Sem Fio (RSMSF), definida basicamente pela interconexão de nós sensores dispostos de forma distribuída e capazes de monitorar um ambiente através do uso de múltiplas mídias (Akyildiz & Vuran 2010). Desta forma, é possível não só que haja a captura de áudio e vídeo mas também a manipulação dos dados obtidos a partir dessas fontes, de forma que possam ser armazenados, processados, correlacionados e fundidos a fim de gerar informações relevantes. Com o surgimento das RSMSF novos desafios também são introduzidos em diversas áreas como processamento de sinais, comunicação, redes, controle e estatística. Dentre as áreas de aplicação mais estudadas estão aquelas relacionadas ao controle de tráfego, à vigilância multimídia e assistividade, que particularmente vem recebendo bastante atenção. Este trabalho tem como objetivo estabelecer uma infraestrutura de rede de sensores multimídia sem fio capaz de oferecer serviços para um

robô móvel presente no ambiente. No caso em questão, duas aplicações são apresentadas, uma de auxílio à navegação e outra de localização, ambas responsáveis por assistir a navegação do robô. A principal motivação do trabalho são ambientes assistivos para auxílio a robôs móveis semi-autônomos conduzindo pessoas com severas limitações motoras. Nas aplicações desenvolvidas os nós de uma RSMSF realizam a captura de imagens e fazem uso de técnicas de visão computacional para extrair dados relevantes à navegação e localização do robô. Esses dados são processados na própria rede e as informações geradas são transmitidas através da utilização de um algoritmo de roteamento desenvolvido durante o trabalho. As mensagens são recebidas e agregadas pelo nó anexado ao robô, e então encaminhadas ao robô a fim de informá-lo sobre sua pose ou sobre alguma ação que deve ser tomada em função do estado do ambiente monitorado.

Esta dissertação está organizada em quatro capítulos: o Capítulo 1 apresenta conceitos relacionados às RSSF e introduz as RSMSF, além de trabalhos correlatos; o Capítulo 2 apresenta a infraestrutura de rede desenvolvida assim como as ferramentas e procedimentos adotados para a sua implementação; o Capítulo 3 apresenta os serviços de auxílio à navegação e de localização implementados; o Capítulo 4 apresenta os testes e experimentos realizados, além dos resultados obtidos para validação do sistema; e, por fim, são apresentadas as conclusões e propostas de trabalhos futuros. Foi adotada a convenção de manter alguns termos técnicos em inglês a fim de manter a fidelidade semântica com o seu significado na área.

Capítulo 1

Redes de Sensores Sem Fio

Sensores são dispositivos capazes de capturar grandezas associadas a um determinado fenômeno físico e convertê-las em sinais que podem ser processados, transmitidos e armazenados (Dargie & Poellabauer 2010). Esses dispositivos podem estar integrados em máquinas, ambientes, dentre outros locais, desempenhando papéis de extrema importância como na prevenção de falhas em infraestruturas, na conservação de recursos naturais, no aumento da produtividade industrial, na segurança e, em aplicações mais recentes, envolvendo sistemas sensíveis a contexto e ambientes inteligentes.

O grande avanço nas áreas da micro-eletromecânica, comunicação sem fio e eletrônica digital tornou possível a miniaturização do *hardware* dos sensores sem comprometer suas capacidades de processamento e armazenamento. A redução do tamanho e, consequentemente, do consumo dos componentes, acompanhada pela diminuição da quantidade de material necessária para manufatura dessa tecnologia, contribuiu para a redução do custo, viabilizando o uso de sensores em aplicações de grande porte. Esse capítulo irá abordar os conceitos, definições e convenções da área de RSSF, além de apresentar trabalhos relacionados ao desenvolvido nesta pesquisa.

1.1 Definição de RSSF

As RSSF são consideradas uma subclasse das redes *ad hoc* móveis (MANET - *Mobile Ad Hoc Networks*). Esse tipo de rede é utilizada para atender propósitos específicos e é capaz

de suprir a necessidade de comunicação de forma rápida e prática. Nesse modelo cada nó se comunica com os outros colaborativamente, o que possibilita a extensão da área de sensoriamento que passa a abranger a própria área de cobertura da rede (Karl & Willig 2005). Apesar das semelhanças, as redes *ad hoc* convencionais e as RSSF possuem diversas características que as diferem, principalmente no que diz respeito às aplicações, equipamentos, consumo de energia, mobilidade e qualidade de serviço.

Uma RSSF é a interconexão de dispositivos eletrônicos de baixo custo chamados de nós sensores (conhecidos também como *motes*), utilizados basicamente para realizar a coleta de dados em uma área de observação. Após a coleta, os dados obtidos pelos nós são transmitidos, de nó a nó (mecanismo de múltiplos saltos), até alcançar a entidade que os solicitou, chamada de nó Servedouro, onde os dados podem ser processados, visualizados, analisados e armazenados (Pottie 2001), como pode ser visto na Figura 1.1.

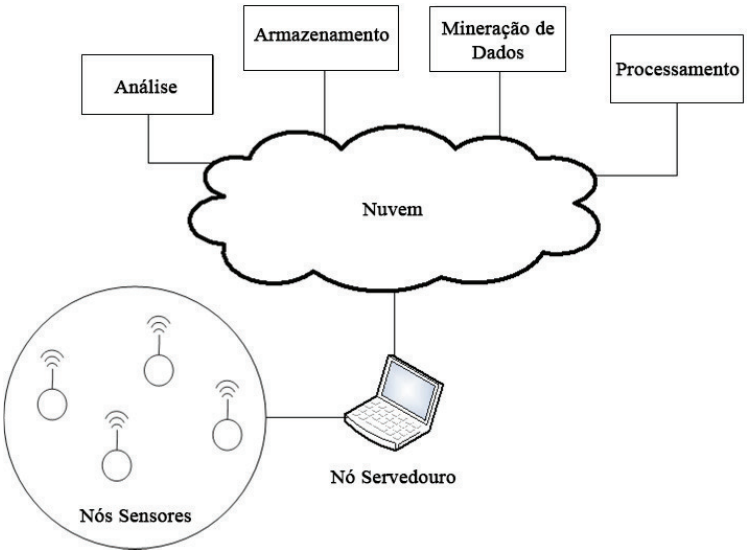


Figura 1.1: Arquitetura típica de uma rede de sensores sem fio.

O nó Servedouro pode se apresentar em uma RSSF de três maneiras: como um nó sensor ou atuador, como uma entidade externa usada para interação com a rede (como computadores portáteis, por exemplo), ou simplesmente como um *gateway* para redes maiores, como a Internet. Em resumo, um nó sensor não se trata apenas de um componente de sensoriamento com capacidades de comunicação, mas também apresenta capacidades de processamento e armazenamento

que permitem o tratamento dos dados capturados do meio. A Figura 1.2 mostra a arquitetura básica de um nó sensor.

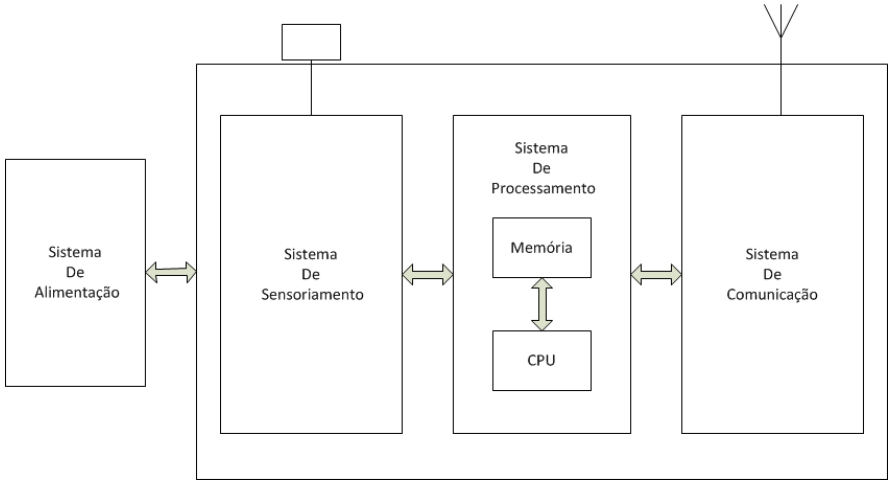


Figura 1.2: Arquitetura básica de um nó sensor.

Essas capacidades fazem com que esses dispositivos sejam capazes de realizar coleta de dados, análises *in-network*, correlação e fusão de seus próprios dados com os obtidos a partir de outros nós sensores (Dargie & Poellabauer 2010). A principal característica desse tipo de rede é a cooperação para realizar uma tarefa comum, sendo bastante útil quando a aplicação exige uma grande cobertura de conectividade ou quando a área a ser observada se trata de um local remoto e/ou de difícil acesso.

Uma das grandes vantagens das RSSF é a mobilidade que ela proporciona para a aplicação. Essa mobilidade gera uma alta dinamicidade na rede, que pode tanto crescer como diminuir a qualquer momento. Em uma área de observação, essa característica pode ser abordada de três maneiras dentro do contexto da aplicação. No primeiro caso os nós sensores são móveis, ou seja, a rede precisa se reorganizar frequentemente para que haja a convergência necessária para seu correto funcionamento. Muitas vezes essa mobilidade não é desejada, como por exemplo na extração de dados estatísticos, o que exige o monitoramento contínuo da mesma área de observação. No segundo caso, o nó Servedouro apresenta mobilidade. Este cenário, apesar de incomum em RSSF, vem sendo considerado em ambientes inteligentes e assistivos. Neste caso, o nó Servedouro pode ser um computador de mão controlado por um usuário que percorre o

ambiente, assim como também pode ser uma entidade autônoma, como um robô móvel, por exemplo. Dessa forma, essas entidades móveis podem realizar requisições para a rede de acordo com a tarefa às quais foram designadas. No terceiro caso, o próprio evento ou objeto que está sendo monitorado apresenta mobilidade (Karl & Willig 2005). Nesse caso é necessário que a rede proporcione cobertura suficiente para a detecção do evento, por isso é normalmente utilizado em aplicações de ambientes fechados. Uma aplicação de RSSF pode apresentar vários tipos de mobilidade de acordo com sua necessidade. As aplicações desenvolvidas neste trabalho apresentam mobilidade no evento e no nó Servedouro. Essas aplicações serão abordadas com detalhes no Capítulos 3 desta dissertação.

A flexibilidade proporcionada pela mobilidade em RSSF, onde basta um nó ser ligado e introduzido na área de cobertura da rede para fazer parte dela, faz com que naturalmente a arquitetura de uma RSSF seja desestruturada. Uma arquitetura desestruturada é uma coleção de nós sensores espalhados, geralmente de maneira aleatória. Este tipo de infraestrutura inviabiliza a manutenção e o gerenciamento dos nós, que geralmente são encontrados em grande quantidade devido a facilidade de implantação. Apesar desse tipo de arquitetura ser comumente encontrada, aplicações recentes vêm fazendo uso de infraestruturas controladas, onde os nós são implantados de forma pré-planejada, simplificando a manutenção e gerenciamento dos dispositivos.

1.2 Características Básicas das RSSF

Embora as RSSF sejam semelhantes aos outros sistemas distribuídos, elas apresentam algumas restrições e desafios únicos. Essas restrições interferem diretamente no projeto e leva ao desenvolvimento de algoritmos e protocolos diferentes dos usados nos sistemas convencionais (Gutierrez, Naeve, Callaway, Bourgeois, Mitter & Heile 2001). A principal restrição de uma RSSF é a limitação de energia. Normalmente os nós sensores são alimentados por baterias que devem ser trocadas ou recarregadas. Contudo, em algumas aplicações nenhuma dessas opções é apropriada, ou seja, o nó é simplesmente descartado quando sua energia se esgota. Isso ocorre em aplicações que faz uso de uma quantidade muito grande de nós, geralmente posici-

onados em locais de difícil acesso, tornando inviável a troca de baterias. Com essa limitação, o tempo que a rede, ou uma parte dela, vai ficar operante também se restringe. Neste caso, o projeto de RSSF deve ser realizado utilizando técnicas para otimizar o consumo de energia a fim de prover tempo suficiente para a rede cumprir seus objetivos.

Dentre os componentes que compõe um nó sensor, o transceptor (sistema de comunicação) é considerado o mais importante, pois é ele quem consume maior quantidade de energia e quem provê conectividade para o resto da rede. Quase todos os dispositivos comerciais utilizam transmissão via radiofrequência (RF) como padrão. Essa tecnologia embora consuma bastante energia para transmitir dados a longas distâncias, é considerada eficiente em termos de consumo energético na troca de dados em curtas distâncias (Hill, Horton, Kling & Krishnamurthy 2004). Sua utilização possibilita o envio eficiente de pacotes pequenos usando baixas taxas de transmissão de dados. Adicionalmente, a tecnologia de RF permite o desenvolvimento de operações de *duty-cicle*, ou seja, operações que permitem ao transceptor ficar desligado durante a maior parte do ciclo de vida do nó sensor e ser ligado apenas se houver um pacote a enviar ou receber. Apesar de acrescentar vantagens às RSSF, o seu uso requer modulação, filtragem, demodulação e multiplexação, o que torna o nó mais complexo e demanda consumo de energia. A comunicação por radiofrequência será explicada com mais detalhes no decorrer deste capítulo.

Devido ao fato dos nós sensores serem circuitos de baixo custo, eles estão mais suscetíveis a interferências e a danos físicos. Além disso, devido suas limitações de memória e baixa capacidade de processamento, são necessárias adequações e otimizações de *software* para que não ocorram sobrecargas de memória e de processamento capazes de interromper a operação do nó. Essas falhas podem tornar os nós inoperantes por tempo indeterminado, o que pode resultar na inviabilização da rota que interconecta o nó Servedouro ao resto da rede, tornando a RSSF incapaz de cumprir seu propósito. Diante disto, uma RSSF deve apresentar mecanismos para que a perda de uma pequena quantidade de nós não interfira no seu funcionamento geral. Esta capacidade que uma rede tem de se manter funcional, ou seja, sem interrupções operacionais devido à falhas em nós sensores é denominada tolerância à falhas (Akyildiz, Su, Sankarasubramaniam & Cayirci 2002). A tolerância à falhas em uma RSSF depende de vários fatores. Um

deles é o local onde a rede será implantada. Se esse ambiente tiver poucas interferências, então os protocolos podem ser mais flexíveis devido a baixa probabilidade de ocorrência de falhas. Sendo assim, podemos considerar que a tolerância à falha depende da aplicação em questão, ou seja, o tipo de ambiente em que vai ser implantada, quais são os pontos críticos do sistema para ocorrência de falhas, qual o tempo estimado para que a RSSF cumpra seu objetivo, etc. Uma das principais estratégias para criar um sistema tolerante à falhas é criar redundância de nós, ou seja, colocar vários nós na área de abrangência de um outro nó (Akyildiz & Vuran 2010). Essa redundância, no entanto, apesar de apresentar vantagens, aumenta significativamente a densidade da rede, o que introduz outros desafios relacionados ao projeto de uma RSSF.

1.3 Plataformas de *Hardware*

Um nó de uma RSSF é formado por um conjunto de sistemas embarcados que juntos permitem a interação com o ambiente, o processamento e transmissão de informações. Nas soluções comerciais, esses sistemas podem estar acoplados a um único dispositivo físico ou podem ser divididos em módulos que podem ser anexados a um módulo principal. No geral, os módulos mais comuns são o de comunicação, de sensoriamento e de programação. O módulo de comunicação é considerado o módulo principal de uma plataforma de RSSF, pois é ele quem aloja o processador, a memória, a fonte de energia e o transceptor. O módulo de sensoriamento agrega vários tipos de sensores escalares e muitas vezes oferece interfaces para que outros tipos de sensores possam ser incluídos. O módulo de programação, por sua vez, é aquele que oferece uma interface através da qual o desenvolvedor pode se conectar ao módulo principal e descarregar códigos de aplicação ou transferir algum tipo de dado.

Além dos módulos citados, que são bastante comuns em RSSF, soluções mais recentes fornecem outros tipos, como módulos multimídia, por exemplo, compostos de câmeras e microfones. Com o uso desse tipo de tecnologia, que lida com um maior volume de dados, tornou-se necessário o aumento das capacidades de processamento e armazenamento das plataformas de RSSF a fim de lidar com os diferentes tipos de dados gerados a partir dessas fontes. Com base neste avanço, as plataformas de RSSF passaram a ser dividas em dois grupos: *low-end* e *high-end* (Margi,

Petkov, Obraczka & Manduchi 2006). As plataformas *low-end* se caracterizam basicamente pela baixa capacidade de processamento e armazenamento, além de limitações na comunicação. Esses dispositivos são produzidos com microcontroladores e transceptores de baixo consumo, o que diminui o custo e permite que sejam empregados em larga escala, principalmente para realizar tarefas de sensoriamento e para proporcionar uma infraestrutura de conectividade. A maioria dos protocolos de comunicação é desenvolvida utilizando esses dispositivos mais simples devido a viabilidade para realização de testes de escalabilidade. Alguns exemplos deles são os *motes* da família Mica, o Telos/Tmote e o EYES.

As plataformas *high-end* se diferenciam por terem melhores capacidades em relação aos dispositivos *low-end*. Essas plataformas são geralmente utilizadas em aplicações que requerem mais recursos, como no gerenciamento de rede e no processamento de dados coletados, por exemplo. Elas também são bastante utilizadas como *gateways* para integrar infraestruturas já existentes. Como seu custo é mais elevado e seu consumo de energia é maior, essas plataformas são utilizadas em pequenas quantidades no sistema. Alguns exemplos são o Stargate e o IntelMote2. A Tabela 1.1 apresenta as principais plataformas comerciais e descreve suas características (Souza 2011).

Tabela 1.1: Plataformas de Hardware.

Nome	Catg.	Vel. da CPU	Mem. de Prog.	RAM	Tx. Transf.
MicaZ	low-end	16MHz	128kB	4kB	40kbps
TelosB/Tmote	low-end	16MHz	48kB	10kB	250kbps
SHIMMER	low-end	8MHz	48kB	10kB	250kbps
Sun SPOT	low-end	16-60MHz	2MB	256kB	250kbps
Imote	low-end	12MHz	512kB	64kB	100kbps
Imote2	high-end	13-416MHz	32MB	256kB	250kbps
Stargate	high-end	400MHz	32MB	64MB SD	Variável

O tipo de plataforma utilizada em um aplicação de RSSF é escolhida ainda na fase de projeto, dependendo do tipo da aplicação e do cenário em que a rede será implantada. Apesar das plataformas descritas na Tabela 1.1 serem específicas para RSSF, diversas empresas trabalham no desenvolvimento de plataformas bastante poderosas compostas de co-procesadores de alta capacidade para tratamento de dados multimídia, possibilitando a criação de sistemas de controle e monitoramento sofisticados.

1.4 Sistemas Operacionais para RSSF

Uma plataforma de RSSF é formada por diversos elementos como processador, memória, antenas de rádio, sensores, etc. O sistema operacional provê uma interface de *software* localizada logicamente entre esses componentes de *hardware* e as aplicações de usuário, responsável principalmente por fornecer abstrações básicas de programação, ou seja, uma interface para acesso ao *hardware* da plataforma (Dargie & Poellabauer 2010). Sua principal função é gerenciar processos e recursos a fim de organizar e otimizar o funcionamento do dispositivo. Em RSSF, devido a escassez de processamento e armazenamento, e memória RAM (*Random Acess Memory*) limitada, os sistemas operacionais são bastante simplificados. Além disso, pelo fato das plataformas apresentarem restrições, relacionadas principalmente com requisitos voltados para economia de energia, os sistemas operacionais devem ser desenvolvidos para operarem da forma mais eficiente possível quanto ao consumo de cada componente.

Os sistemas operacionais para RSSF podem ser classificados de acordo com os modelos de programação, que podem ser: baseado em multitarefas e baseado em eventos. O modelo de programação baseado em multitarefas permite a execução de múltiplos processos de forma concorrente em uma única CPU (*Central Processing Unit*), Figura 1.3, e é suportado por uma grande variedade de CPUs. Apesar desse modelo ser considerado o mais natural para implementação, ele apresenta diversos obstáculos relacionados às restrições das RSSF. Nesse esquema, todas as tarefas são processadas enquanto estiverem ativas, porém de forma preemptiva onde cada uma é processada em intervalos de tempo diferentes, que são alternados para que todas sejam atendidas pelo sistema operacional. Isso, além de gerar uma falsa impressão de paralelismo, acarreta em uma grande sobrecarga no dispositivo, pois cada tarefa exige um espaço em memória para armazenamento de seus estados. Dessa forma, o aumento do número de tarefas no escalonador da CPU diminui consideravelmente o seu desempenho. Uma alternativa é a execução sequencial de tarefas onde a tarefa executa até terminar (sem preempção).

Já no modelo de programação baseado em eventos, Figura 1.4, o sistema basicamente espera pelo acontecimento de um evento, que pode ser a chegada de um pacote de rede, por exemplo. Para cada evento é registrado um *handler*, uma função que executa quando há a ocorrência

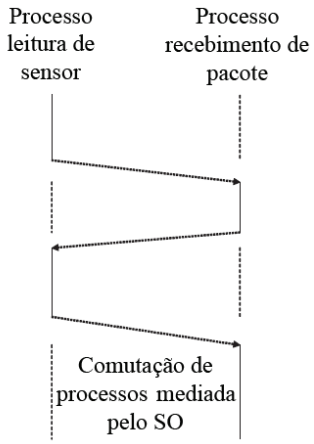


Figura 1.3: Modelo baseado em multitarefas.

do evento. O modelo baseado em eventos apresenta uma série de vantagens, entre elas, a diminuição do consumo de energia e aumento do desempenho, pois além de passar grande parte do tempo ocioso, aguardando pelo acontecimento de eventos, não realiza processamento desnecessário. Apesar de sistemas baseado em eventos terem sido muito utilizado nos últimos anos, recentemente os desenvolvedores têm investido esforços em sistemas baseados em multitarefas, porém fazendo uso de implementações mais leves e otimizadas (Farooq & Kunz 2011).

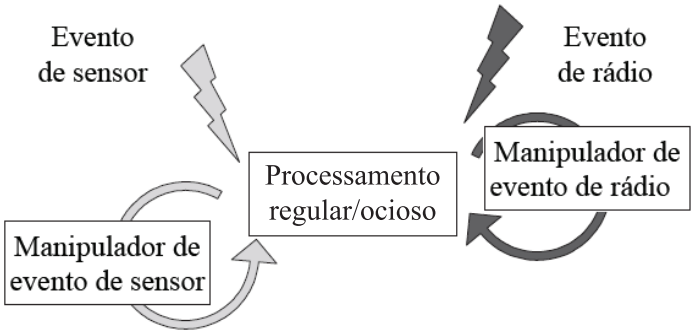


Figura 1.4: Modelo baseado em eventos (Karl & Willig 2005).

O sistema operacional adotado nesse trabalho foi o Linux. Ele é um sistema em constante evolução e que passou a ser utilizado também em RSSF, além de ser bastante conhecido e com uma grande comunidade de desenvolvedores. Ele já vem sendo utilizado em sistemas embarcados há alguns anos e já foi portado para um grande número de CPUs. Além de ser robusto e flexível, ele dispõe de uma grande quantidade de *drivers* e protocolos já implementados. Apesar de suas

qualidades, por apresentar uma sobrecarga de processamento maior em comparação aos outros sistemas para RSSF existentes, ele é utilizado em pequena escala, geralmente em dispositivos *high-end*. As vantagens do Linux em sistemas embarcados estão na flexibilidade de acesso ao código-fonte, na sua popularidade e no rico conjunto de ferramentas e aplicações disponíveis, permitindo o desenvolvimento de sistemas complexos.

Em plataformas *low-end* o sistema mais bem aceito é o TinyOS (Akyildiz & Vuran 2010). Ele possui código aberto e incorpora uma arquitetura baseada em componentes, minimizando o tamanho do código e proporcionando flexibilidade para o desenvolvimento de protocolos de comunicação. Sua biblioteca de componentes inclui protocolos de rede, serviços distribuídos, *drivers* de sensores e ferramentas de aquisição de dados, que podem ser modificadas ou melhoradas para atender as necessidades de aplicações específicas. Seu projeto inicial é baseado em um modelo de execução dirigido a eventos o que possibilita boas estratégias para o gerenciamento de energia.

Além do Linux e do TinyOS, diversos outros sistemas operacionais vêm sendo utilizados em RSSF. Os principais são listados e descritos abaixo:

- LiteOS: é um sistema operacional multitarefa baseado em Unix. Oferece suporte a programação orientada a objeto através da linguagem LiteC++ e possui uma interface com o usuário através de um ambiente de linha de comando chamado LiteShell. Ele é bastante leve e pode ser usado até mesmo em dispositivos bastante limitados como o MicaZ, por exemplo (Cao, Abdelxaher, Stankovic & He 2008).

- O Contiki é um sistema de código aberto feito para diversas plataformas e escrito em linguagem C . Ele foi desenvolvido sobre um *kernel* dirigido à eventos porém também oferece suporte a multitarefas. Além disso, ele possui suporte a TCP/IP (*Transport Control Protocol/Internet Protocol*), IPv6 (*Internet Protocol Version 6*), interface gráfica com o usuário, navegador de Internet, servidor *Web*, cliente Telnet, dentre outros recursos (Dargie & Poellabauer 2010).

- O MANTIS é um sistema operacional multitarefa de código aberto e escrito em linguagem C. Ele foi desenvolvido especificamente para plataformas *low-end* por isso apresenta baixo

consumo de energia e baixo consumo de memória RAM (*kernel*, agendador de tarefas e pilha de rede não ultrapassam 500 Bytes em RAM). Para economizar energia, o MANTIS realiza o desligamento do controlador em casos que não há demanda de comunicação de dados. Apesar de ser bastante simplificado, o sistema operacional oferece suporte ao gerenciamento remoto e depuração via PC (*Personal Computer*) (Bhatti, Carlson, Dai, Deng, Rose, Sheth, Shucker, Gruenwald, Torgeson & Han 2005).

- O Nano-RK é um sistema operacional multitarefas para aplicações de tempo real. Ele suporta agendamento de tarefas baseado em prioridade, possui tempo de vida estendido, cota para utilização de recursos e baixo consumo de memória RAM (apenas 2kB). O sistema também oferece suporte a rede através de uma abstração baseada em *sockets* (Farooq & Kunz 2011).

1.5 Pilha de Protocolos para RSSF

Em RSSF, normalmente, um nó só é capaz de encaminhar mensagens diretamente aos seus vizinhos, ou seja, para aqueles nós que estão dentro da área de cobertura do seu sinal de rádio. Nas plataformas de sensores existentes esse alcance é bastante limitado devido às características do *hardware*, que é projetado visando a redução do valor comercial do dispositivo e minimização do consumo de energia. Contudo, como uma RSSF deve ser capaz de atuar em áreas extensas, é necessário que os protocolos de comunicação operem de um modo que possibilite a chegada de uma mensagem até o nó Servedouro, que pode estar afastado da área de alcance do nó originador da mensagem. Um nó de uma RSSF pode ser classificado quanto a sua função, em duas categorias: gerador de dados e roteador de dados. A função de gerar dados é exercida quando um nó detecta um determinado evento através de sensoriamento e o transmite para a rede através de difusão. Já a função de roteamento é executada quando um nó recebe um pacote, verifica que não é o destinatário e por isso o encaminha adiante até que, em algum momento, ele seja recebido pelo nó ao qual foi destinado. O mecanismo de roteamento permite que a mensagem trafegue de nó em nó através de múltiplos saltos até alcançar o destino, que normalmente

é o próprio nó Servedouro. Ao receber a mensagem, o nó Servedouro se responsabiliza por disponibilizar as informações sensoriadas ao usuário final, seja pela Internet, utilizando redes sem fio (Wifi, celular, WiMax, etc.), ou até mesmo através de uma conexão direta, via cabo serial por exemplo. A pilha de protocolos em RSSF combina métodos de comunicação de baixo consumo de energia que favorecem os mecanismos citados e permitem o acesso dos dados da camada de rede através de chamadas a partir da camada de aplicação. Normalmente, as plataformas comerciais de RSSF oferecem uma pilha de protocolo composta apenas das camadas física e de acesso ao meio, o que possibilita ao desenvolvedor a criação de seu próprio protocolo de camada de rede. Essas três camadas serão descritas a seguir com base nas características da plataforma Imote2, que foi utilizada neste trabalho. As camadas de transporte e aplicação são foco de pouco estudo em RSSF e, apesar de não estabelecidas, serão citadas brevemente.

1.5.1 Camada Física

A camada física é responsável pela conversão de um fluxo de bits em um sinal adequado para ser transmitido através de um meio sem fio. Nessa camada é feita a seleção da frequência, geração da portadora, detecção, modulação e codificação do sinal. Quanto mais sofisticadas forem as técnicas empregadas nas etapas citadas, mais confiável será a comunicação, mas também é necessário que características do *hardware*, como sensibilidade da antena e circuito do transceptor, sejam adequadas. O principal desafio no projeto de protocolos de camada física em RSSF é a definição de um meio termo entre as técnicas adotadas e a qualidade da comunicação, ou seja, o ideal é que esses elementos sejam simples e de menor custo possível mas ainda assim sofisticados o suficiente para estabelecer uma comunicação eficiente e robusta.

Grande parte das vantagens oferecidas pelas RSSF está relacionada ao meio sem fio. Devido a natureza desse meio, as RSSF são facilmente implantáveis, livres de infraestrutura e capazes de se comunicar através de difusão. Apesar dessas vantagens, a comunicação sem fio também é responsável por importantes restrições relacionadas à camada física em RSSF. Dentre elas, as mais comuns são: limitação do intervalo de comunicação, alta taxa de erros de transmissão, sombreamento, perda de caminho e interferência. Embora esses problemas mereçam atenção,

eles não diminuem a popularidade da tecnologia, que vem se consolidando cada vez mais.

Em geral, a comunicação sem fio em RSSF pode ser ótica, acústica, por indução magnética ou por RF (Akyildiz & Vuran 2010). A tecnologia de RF é a mais utilizada e a maioria das plataformas de RSSF atuais são baseadas nela. Esse tipo de comunicação ocorre através do uso de ondas eletromagnéticas que são transmitidas sobre bandas de radiofreqüência, que variam no espectro entre 3kHz a 300GHz. As principais técnicas usadas são classificadas em três tipos: *narrow-band*, *spread-spectrum* e *ultra-wide-band*.

A técnica de *narrow-band* visa otimizar a eficiência da largura de banda usando esquemas de modulação M-ário em uma banda estreita. Ela é bastante utilizada pelas plataformas mais antigas de RSSF, como o Mica2, por exemplo, que utiliza o transceptor CC1000 capaz de operar em 443, 868 e 915 MHz, com largura de banda acima de 175 kHz e taxa de transferência de dados de 76 kbps. Diferentemente da *narrow-band*, a *spread-spectrum* propõe uma melhoria na taxa de transferência de dados e na resistência à interferência. Nesse caso, um sinal de largura de banda limitado é espalhado em uma banda mais larga utilizando técnicas de espalhamento de spectrum. Os dois tipos mais conhecidos de espalhamento de espectro são: *frequency hopping spread spectrum* (FHSS) e *direct sequence spread spectrum* (DSSS). Essa segunda é a técnica de espalhamento de espectro padrão em RSSF da atualidade e é empregada em diversas plataformas de 250 kbps, como Imote2 por exemplo. Outra que também vem sendo bastante usada em RSSF é a *Ultra-wide-band*. Esse modelo apresenta um espectro de ampla atuação, permitindo que a transmissão seja feita através de rajadas de sinais (centenas por segundo). Com isso é possível alcançar altas taxas de transmissão e minimizar o consumo de energia devido a menor quantidade de transmissões necessárias no envio de uma mensagem. Maiores detalhes sobre as técnicas citadas podem ser encontrados em (Akyildiz & Vuran 2010).

Na comunicação RF são realizadas as seguintes operações entre o transmissor e o receptor:

- Codificação de fonte (compressão de dados): a fonte de informação (transmissor) é codificada por um codificador de fonte, que explora as estatísticas da informação para representá-la através de um número reduzido de bits. A codificação ou decodificação de fonte é executada na camada de aplicação.

- Codificação de canal (codificação de controle de erro): O código enviado pela fonte é codificado pelo codificador de canal para que os erros causados pelo meio sem fio sobre a informação transmitida possam ser tratados. Para isso, são utilizados códigos corretores de erros.

- Intercalação e modulação: Os símbolos do canal codificado são intercalados para que os erros que possam afetar um número grande de bits consecutivos sejam minimizados. A codificação do canal e a intercalação ajudam o receptor a identificar os erros nos bits para que possam ser retransmitidos ou corrigidos. Em seguida, o sinal é modulado em informação digital e enviado pelo canal através da antena, até o receptor.

- Propagação do canal sem fio: A onda eletromagnética é transmitida e se propaga através do canal. Durante esse processo ela é atenuada e destorcida devido aos efeitos do canal sem fio.

A Figura 1.5 apresenta através de um diagrama de blocos a sequência de operações realizadas durante a comunicação RF.

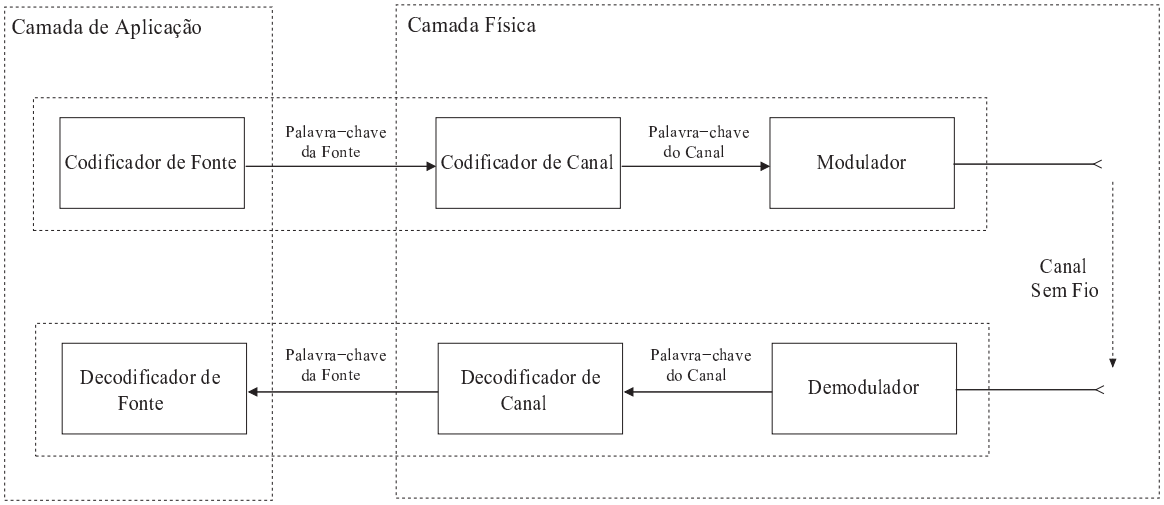


Figura 1.5: Diagrama de blocos de uma comunicação RF (Akyildiz & Vuran 2010).

O sucesso da transmissão por RF depende diretamente dos efeitos do canal e dos parâmetros de operação, como frequência, propriedades da antena e ruído do ambiente. Além disso, a

escassez de energia e os recursos limitados das RSSF tornam necessária a utilização de técnicas de comunicação de baixa complexidade pois quanto mais sofisticados forem os esquemas de modulação e tecnologias de antenas, mais inviável será o sistema de comunicação da plataforma de RSSF.

Com o rápido avanço das RSSF, uma grande variedade de plataformas independentes surgiram nos últimos anos. Para que houvesse interoperabilidade entre elas, o *Institute of Electrical and Electronics Engineers* (IEEE) desenvolveu um padrão para comunicação entre dispositivos de baixo custo chamado IEEE 802.15.4. Esse modelo foi adotado como padrão para RSSF e passou a ser empregado pela maioria das plataformas comerciais e aplicações industriais. O IEEE 802.15.4 foi desenvolvido, mais precisamente, para especificar tecnologias de transceptores sem fio de baixa taxa de transferência de dados com tempo de vida de bateria longo e baixa complexidade. O padrão define as camadas física e de acesso ao meio além de prover flexibilidade para soluções de camadas superiores. Mais especificamente, ele define o espectro sem fio, as técnicas de comunicação e os algoritmos de acesso ao meio que deverão ser usados. Isso permite a compatibilidade de comunicação entre transceptores de diferentes fabricantes. Na camada física, três bandas podem ser usadas para comunicação: 2.4 GHz (global), 915 MHz (América) e 868 MHz (Europa). Além disso, 47 canais são distribuídos entre essas bandas, dentre eles, 16 são reservados para a banda de 2.4 GHz. Nessa faixa, assume-se que o alcance da transmissão dos nós está entre 10 e 100 metros, com taxa de transferência de dados que varia de 20 a 250 kbps.

1.5.2 Camada de Controle de Acesso ao Meio

O canal sem fio apresenta um comportamento caracterizado pela difusão onde o sinal transmitido por um nó sensor possa ser recebido por múltiplos nós sensores ao seu redor. Essa característica é evidenciada pois todo nó sensor compartilha um meio comum a todos os outros que se encontram em sua área de alcance de transmissão. Essa questão é tratada pelos protocolos de camada de controle de acesso ao meio (MAC) que são responsáveis por garantir a comunicação no meio sem fio de tal forma que a conexão entre os nós seja estabelecida, provendo

conectividade à rede como um todo. Para que isso ocorra, o acesso ao meio deve ser coordenado de tal forma que as colisões, decorrentes do acesso simultâneo de nós vizinhos à rede, sejam minimizadas. Os protocolos desta camada podem ser classificados, de acordo com a técnica que utiliza, em dois grupos: os que se baseiam em contenção de acesso e aqueles que usam reserva de canal. Essas técnicas são a base para dois esquemas de acesso fundamentais, o CSMA (*Carrier Sense Multiple Access*) e o TDMA (*Time Division Multiple Access*).

Nas soluções baseadas em reserva cada nó opera em um intervalo de tempo diferente daquele alocado para os demais, por isso esse método é considerado livre de colisões. Já nas soluções baseadas em contenção, geralmente, cada nó realiza a escuta do canal antes de efetuar a transmissão. Caso haja alguma atividade no meio, ou seja, seja detectada a presença de tráfego, indicando que o canal está ocupado, o emissor irá aguardar por um tempo aleatório para então tentar enviar a mensagem novamente. O mecanismo de contenção é mais flexível, pois com ele, cada nó pode executar decisões independentemente, sem precisar realizar troca de mensagens visando o sincronismo, como ocorre no caso do mecanismo de reserva. Por isso, o seu uso não exige que haja uma arquitetura estruturada. Por outro lado, se a rede se tornar muito densa, a probabilidade de colisões aumenta consideravelmente devido ao maior número de nós que irão tentar acessar o meio simultaneamente. Embora a contenção de acesso seja amplamente utilizada em redes sem fio, ela apresenta um alto consumo de energia devido à constante escuta ao meio. Além disso, quando o volume de colisões aumenta, devido ao crescimento da rede, ocorre um aumento da quantidade de retransmissões necessárias para entrega da mensagem e, consequentemente, do consumo de energia. Para estender o uso dessas técnicas para RSSF, adaptando-as às restrições da tecnologia, alguns protocolos de camada MAC foram desenvolvidos, dentre eles os principais são o S-MAC, o B-MAC e o CC-MAC (Akyildiz & Vuran 2010).

O B-MAC é o protocolo nativo das principais plataformas comerciais, como o MicaZ, TelosB e Imote2. A Universidade de Berkeley o desenvolveu com o objetivo de prover um protocolo de camada MAC simples e facilmente configurável. Ele oferece um mecanismo CSMA básico, além de oferecer um mecanismo opcional de confirmação (ACK) no nível de conexão, que elimina a necessidade de troca de mensagens de controle. O protocolo é baseado em dois mecanismos:

o agendamento adormece-desperta, usando escuta de baixo custo, e a detecção de portadora, usando avaliação da ocupação do canal. Os dois mecanismos melhoram a eficiência energética e a utilização do canal. Adicionalmente, pode se encontrar uma implementação do B-MAC para TinyOS, que provê interfaces simples para que os serviços de camadas superiores possam configurar facilmente as operações de acesso ao meio (Lohier, Rachedi, Livolant & Salhi 2011). Essa característica permite o rápido desenvolvimento de soluções intercamadas. Devido a sua simplicidade o protocolo requer pouco espaço de armazenamento, que é um importante requisito em RSSF.

Além da camada física, o padrão IEEE 802.15.4 também especifica uma camada MAC. O padrão especifica um processo de sincronização baseado em duas etapas independentes: com contenção de acesso e outra livre de contenção (Kinney 2003). Ele assume que a rede tenha um nó coordenador responsável por inicializar esse processo de sincronismo. No período livre de contenção, o coordenador realiza uma difusão de quadros a fim de alocar os intervalos de transmissão de cada nó, definindo a prioridade de transmissão de cada um. No período de contenção os nós disputam o acesso ao meio através do mecanismo de CSMA e, aqueles que encontram o meio desocupado, alocam o canal por um determinado intervalo de tempo. Dessa forma pode se dizer que o padrão IEEE 802.15.4 apresenta uma solução híbrida através de operações baseadas em CSMA e TDMA. Embora esse padrão otimize o consumo de energia, ele requer uma topologia baseada em um nó coordenador.

A camada MAC IEEE 802.15.4 provê comunicação para topologias em estrela (Figura 1.6(a)), malha (Figura 1.6(b)) e de *cluster* baseado em árvores (Figura 1.6(c)). Como parte dessas topologias, dois tipos de dispositivos são definidos: os de funções completas (FFDs) e os de funções reduzidas (RFDs). Os FFDs além de implementarem todas a funcionalidades definidas no padrão e serem capazes de se comunicar com qualquer outros nó da rede, também podem ser usados em qualquer topologia, tanto exercendo as características de coordenador como de roteador. Por outro lado, os RFDs possuem uma implementação muito simples e só pode ser utilizado como parte da topologia estrela. Devido a essa simplicidade, eles não podem exercer papeis como os de coordenador ou roteador. Esses dispositivos são capazes de se comunicar

apenas com os coordenadores da rede.

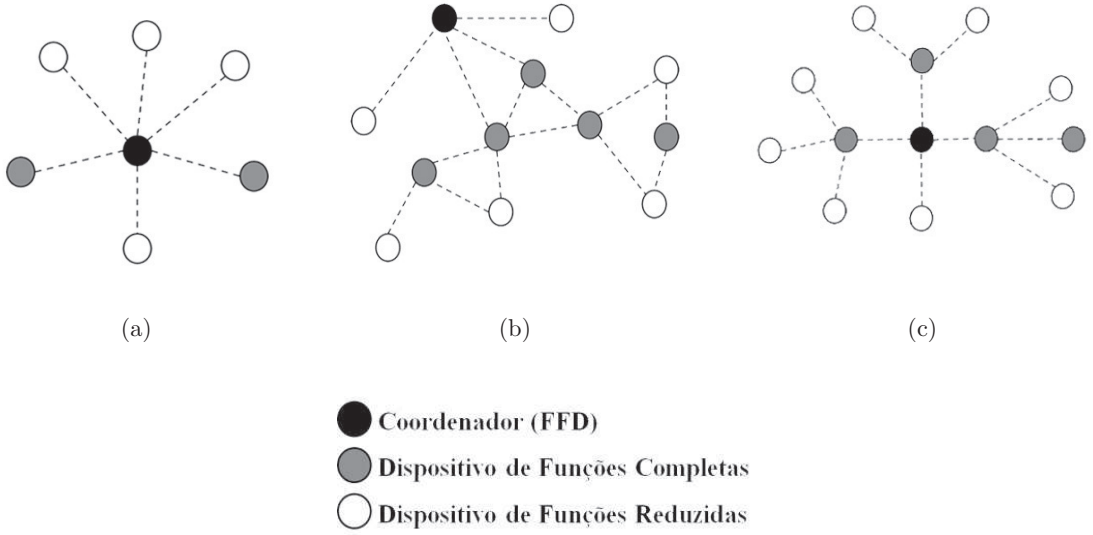


Figura 1.6: Topologias IEEE 802.15.4: (a) Estrela, (b) Malha, (c) Árvore.

1.5.3 Camada de Rede

A camada de rede é responsável por lidar com o roteamento de dados através da rede, a partir do nó originador da mensagem, até um destino, o que talvez envolva a passagem de dados por diversos nós intermediários. O mecanismo de roteamento se resume em duas etapas, a seleção da melhor rota e o encaminhamento da mensagem pela rota encontrada. Esses processos são executados através da difusão de um pedido de rota e da avaliação, a cada salto, de uma ou mais métricas, que são utilizadas para computar o caminho de menor custo.

A métrica mais comum em RSSF é a contagem mínima de saltos. Ela trata basicamente da tentativa de encontrar o caminho do transmissor para o receptor que requer o menor número de saltos. Essa técnica é bastante simples e considera que todo *link* possui o mesmo custo, dessa forma o protocolo de roteamento que a utiliza pode obter o custo total de cada rota e os avaliar comparativamente. A ideia básica por traz dessa métrica é que o uso do menor caminho permite que a mensagem alcance rapidamente o destino através de um baixo consumo de energia

da RSSF, devido ao pequeno número de nós utilizados ao longo da rota. Entretanto, como essa abordagem não considera a disponibilidade corrente de recursos de cada nó, a rota resultante pode ser não-ótima em termos de latência, consumo energético e congestionamento. Além de saltos, outros fatores bastante comuns na definição das métricas são: a energia, a qualidade do serviço e a robustez.

A eficiência energética é o fator mais importante em RSSF, por isso existem vários modos de abordar esse requisito. As métricas mais naturais que tratam esse problema são as que utilizam o consumo mínimo de energia por pacote e a que avalia a capacidade energética dos nós da RSSF. O primeiro caso visa minimizar o consumo total de energia gasta na transmissão de um pacote, da origem até o destino. Esse consumo total é a soma da energia consumida por todos os nós ao longo da rota. Já no segundo caso, é considerada a disponibilidade de energia de cada nó ao longo da rota, ou seja, a rota que possuir nós com maior quantidade de carga disponível, é considerada a de menor custo.

As métricas associadas à qualidade de serviço são definidas de acordo com o desempenho da rede, considerando o atraso na transmissão fim-a-fim, o *throughput* e a perda de pacote, por exemplo. A escolha de métricas dessa categoria depende da aplicação na qual a RSSF irá ser empregada. As redes que executam detecção de alvo e *tracking* requerem baixo atraso para transmissão fim-a-fim pois os dados obtidos são sensíveis ao tempo, enquanto redes de tráfego intenso, como as Redes de Sensores Multimídia Sem Fio (RSMSF), vão requerer alta vazão, por exemplo. Uma métrica bastante comum baseada na qualidade de serviço é o ETT (*Expected Transmission Time*), que é utilizado para expressar a latência. O ETT é definido pela Equação 1.1 (Dargie & Poellabauer 2010) onde S é o tamanho médio do pacote, B é a vazão e ETX (*Expect Transmission Count*) é o número de transmissões necessárias para entrega bem sucedida de um pacote sobre um meio sem fio.

$$ETT = ETX \cdot \frac{S}{B} \tag{1.1}$$

Métricas de roteamento que exploram a robustez são utilizadas em casos que a aplicação requer o uso de rotas estáveis e confiáveis. Uma métrica bastante usada é a estimativa da

qualidade do *link* do nó em relação a seus vizinhos. Através desse método cada nó pode então selecionar o próximo salto de forma eficiente, aumentando a probabilidade de sucesso na transmissão até o destino.

Em RSSF, uma estratégia simples para disseminar uma informação pela rede, de modo que esta alcance um nó destino, é a inundação. Nesse processo, um nó emissor envia um pacote em difusão para todos os seus vizinhos, que o reencaminham para seus vizinhos, e assim sucessivamente. Isso ocorre até que todos os nós da rede tenham recebido o pacote, ou até que o pacote tenha sido transmitido por um número limite de vezes. Esse valor deve ser definido adequadamente de tal forma que todo nó da rede possa ser alcançado, mas que o pacote não trafegue por muito tempo desnecessariamente. A técnica de inundação garante que, se existe um caminho até o nó destino e se esse caminho for alcançável, o nó destino irá receber o dado enviado. A principal vantagem desse método é a sua simplicidade, tendo como desvantagem o tráfego intenso provocado na rede.

Uma variação da inundação é a técnica de *gossiping*. Nessa técnica, ao invés do nó sempre enviar o pacote em difusão, é utilizada uma abordagem probabilística onde a decisão de encaminhar um pacote para seus vizinhos tem probabilidade P e a decisão de descartar o pacote, $1-P$. Dessa forma, é possível reduzir a quantidade de tráfego e, consequentemente, economizar energia através da aleatoriedade. Entretanto, com o uso dessa técnica é possível que a entrega de dados de um sensor falhe caso seu único vizinho decida não encaminhar o dado para ele. Além disso, se uma alta probabilidade para encaminhar for definida, o volume de transmissões se torna alto (a probabilidade 1 corresponde a inundação), diminuindo os benefícios dessa técnica. Por outro lado, se a probabilidade for baixa, a sobrecarga pode ser significativamente menor, porém a probabilidade de entrega de dados ser mal sucedida aumenta.

Os protocolos de roteamento para RSSF podem ser pró-ativos ou reativos. Os pró-ativos mantêm tabelas de roteamento precisas e consistentes em todos os nós da rede, através de disseminações periódicas das informações de roteamento. Nesse modelo as rotas são criadas antes de que o encaminhamento de uma mensagem seja solicitado. Já os reativos são aqueles nos quais as rotas são criadas dinamicamente e sob demanda, ou seja, a partir de uma solicitação

de rota. Existem sete categorias que classificam os protocolos de roteamento para RSSF (Shio Kumar Singh & Singh 2010):

- Protocolos baseados na localização: utilizam a localização dos nós sensores para calcular a distância entre dois nós de modo a estimar a rota de menor custo energético.
- Protocolos centrados em dado: são centrados nos pacotes de dados, ou seja, todo nó pode tomar decisões e atuar sobre o conteúdo do pacote, alterando-o ou agregando algum tipo de informação.
- Protocolos hierárquicos: consideram a subdivisão lógica da rede em grupos, onde cada um é gerenciado por um nó coordenador, dotado de maiores capacidades. Esse nó é responsável por gerenciar as transmissões de dados realizadas pelos nós de seu grupo. Nesse modelo a rede é vista de modo hierárquico através de várias camadas, onde os nós de uma camada inferior só pode reportar ao nó de uma camada superior, que por sua vez irá rotear as mensagens e trocar informações com os outros nós de mesma camada ou superiores, até que o destino seja alcançado.
- Protocolos baseados em mobilidade: operam com base na mobilidade do Servedouro a fim de estimar sua posição e utilizar nós próximos a ele para transmitir os dados. Dessa forma é possível otimizar o uso de energia e garantir a entrega da mensagem (evita a contagem para o infinito, ou seja, o número máximo de saltos permitidos para que um nó seja considerado alcançável).
- Protocolos baseados em múltiplos caminhos: cada nó sensor seleciona os K melhores caminhos e divide a carga de informação por esses caminhos, até que a mensagem alcance o destino.
- Protocolos baseados em heterogeneidade: consideram que existem dois tipos de dispositivos na rede, aqueles alimentados via cabo, que não sofrem restrições de energia, e aqueles alimentados por baterias, que possuem tempo de vida limitado e cuja energia deve ser utilizada de forma eficiente através da otimização da comunicação e do processamento de dados.

- Protocolos baseados em qualidade de serviço: consideram restrições como atraso, confiabilidade e tolerância à falhas.

Os protocolos de roteamento são diferentes daqueles desenvolvidos para redes tradicionais. Uma das principais características a ser destacada é que nós sensores não suportam endereçamento IPv4 (*Internet Protocol version 4*), o que impede o desenvolvimento de protocolos baseados nesse padrão. Outro fator importante, é que o projeto de protocolos de roteamento para RSSF devem ser escaláveis, ou seja, a comunicação deve ser estabelecida de forma eficiente e as mensagens devem ser propagadas até o seu destino independentemente da densidade da rede. Os protocolos também devem considerar as restrições das RSSF, como energia, largura de banda, escassez de memória e capacidades de processamento.

O roteamento é um dos tópicos mais estudados na área de RSSF e uma grande quantidade de protocolos de roteamento são desenvolvidos todos os anos, além daqueles desenvolvidos para redes Ad Hoc que são reaproveitados. Apesar desse volume, poucos deles são implementados ou apresentam eficiência em cenários reais (Forster & Murphy 2010). Além disso, aqueles que não são implementados em cenários reais, mas somente testado em ambientes simulados, fazem uso de ferramentas inadequadas para validar o algoritmo, ou seja, simuladores cuja modelagem não atendem as restrições e características de sistemas de RSSF. Alguns dos protocolos que são largamente utilizados em cenários reais são o CTP (*Collection Tree Protocol*), DYMO (*Dynamic MANET On Demand Protocol*), AODV (*Ad Hoc On Demand Distance Vector Protocol*), DSR (*Dynamic Source Routing Protocol*), DSDV (*Destination-Sequenced Distance Vector Protocol*), TORA (*Temporally Ordered Routing Algorithm*) e DD (*Directed Diffusion*) (Thorup 2007).

Simuladores para Redes de Sensores Sem Fio

Uma das formas mais adotadas para teste e validação de protocolos de roteamento, antes de implementá-los em cenários reais, é a simulação. Ela é utilizada principalmente devido a inviabilidade de se testar cada nova implementação ou modificação, em ambientes com características diferentes. Além disso, com ela é possível minimizar o esforço e o tempo gasto na implantação dos nós que compõe a RSSF. Por essa razão, as ferramentas de simulação são amplamente uti-

lizadas, oferecendo fácil depuração, monitoramento e controle do sistema. Com isso, é possível observar de modo mais criterioso as interações entre os nós, algo que não seria trivial em um cenário real.

As RSSF apresentam restrições que devem ser consideradas na escolha de um simulador. Um requisto essencial é a escalabilidade, pois essa tecnologia de rede, normalmente, é composta de uma grande quantidade de nós. Por isso, esse tipo de ferramenta deve apresentar resultados corretos e precisos o suficiente para que as conclusões obtidas possam ser semelhantes àquelas de um cenário real. Para que isso seja possível o simulador deve oferecer um modelo de bateria e energia, assim como modelos realísticos do ambiente físico e dos mecanismos de propagação do sinal. Adicionalmente, a arquitetura do simulador deve ser flexível de forma que o desenvolvedor possa manipular os parâmetros e atributos da rede de maneira fácil e rápida. No momento, existem diversos ambientes de simulação que podem ser usados em RSSF, entretanto, eles variam consideravelmente no que diz respeito a estrutura e características providas, como modelos e protocolos. A Tabela 1.2 apresenta os simuladores mais comuns e que ainda são mantidos pela comunidade de RSSF, assim como suas respectivas características (Kellner, Behrends & Hogrefe 2010).

Dentre os simuladores mais utilizados, estão aqueles que já eram utilizados para redes convencionais e foram estendidos, a fim de prover suporte para RSSF, e aqueles que foram desenvolvidos especificamente para esta tecnologia. A grande vantagem daqueles que foram estendidos é a grande quantidade de algoritmos já desenvolvidos, que podem ser modificados e reaproveitados para estudos. Já a grande vantagem dos simuladores específicos para a tecnologia é o fato de apresentarem uma modelagem mais consistente com as restrições reais das RSSF.

1.5.4 Camada de Transporte e Camada de Aplicação

Em RSSF, as camadas de aplicação e de transporte normalmente não são focos de muitos estudos quando comparadas as outras, que são críticas para o estabelecimento de uma rede. A camada de transporte é basicamente responsável por executar o controle de congestionamento a fim de minimizar a perda de pacotes, aumentando a confiabilidade na entrega, e por definir

Tabela 1.2: Simuladores para RSSF.

Simulador	Versão Atual	Licença	Linguagem	Características
Qualnet	5.0 (2009)	comercial	C e Parsec	Oferece mobilidade básica, modelo de propagação de rádio 802.11, bateria e modelo de energia
OPNET	16.0 (2009)	comercial	C++	Oferece vários modelos de propagação, 802.11, ZigBee e protocolos MANET; apesar de ter ferramentas poderosas e e GUI, não possue suporte específico para RSSF e é caro.
TOSSIM	2.1.1 (2010)	BSD	NesC	Todo baseado no TinyOS, o que possibilita a exportação de códigos para dispositivos baseados em TinyOS sem qualquer modificação.
OMNeT++	4.0 (2009)	Acadêmico	C++ e NED	Oferece vários *frameworks* que implementam características de RSSF, como o MiXiM, Castalia, etc; possui uma grande base de usuários e seu desenvolvimento é baseado no Eclipse.
NS-2	2.34 (2009)	GPL	C++ e OTcl	Grande quantidade de protocolos disponíveis desenvolvidos pela comunidade de usuários, configuração complexa e difícil entendimento devido a grande quantidade de diferentes implementações feitas pelos usuários.
Avrora	1.7.106 (2008)	BSD	Binários AVR	Específico para programas escritos para microcontroladores AVR, com suporte para Mica2 e MicaZ.
Shawn	N.A (2010)	BSD	C++	Se concentra nas camadas mais baixas; não possui nenhum protocolo específico para RSSF.

a aplicação para a qual se destina a mensagem vinda da camada de rede. Existem muitos protocolos de camada de transporte propostos para RSSF. A maioria das soluções existentes focam no transporte confiável de dados segundo a lógica do protocolo TCP, além de questões

relacionadas a erros de conexão sem fio e mobilidade. Apesar de apresentarem relevância, eles não são adequados para RSSF, pois para que o propósito da camada e as restrições das RSSF sejam atendidas simultaneamente, são necessárias modificações significantes em algoritmos bem estabelecidos, como o próprio TCP, por exemplo. Ou seja, a viabilização desses protocolos, de forma que sejam capazes de prover confiabilidade fim-a-fim, com base em confirmação e retransmissão, exige uma grande sobrecarga de implementação.

O principal papel da camada de aplicação em RSSF é abstrair a topologia física da rede através de serviços para serem utilizados pelas aplicações do usuário. Dessa forma o desenvolvedor tem acesso a uma interface para interação com o ambiente através do nó sensor. As soluções de camada de aplicação existentes geralmente envolvem codificação de fonte ou compressão dados, processamento de consultas e gerenciamento da rede (Akyildiz & Vuran 2010). Alguns exemplos são o SMP (*Sensor Management Protocol*), TADAP (*Task Assignment and Data Advertisement Protocol*) e SQDDP (*Sensor Query and Data Dissemination Protocol*) (Bonifácio 2010).

1.6 RSMSF e Aplicações para Assistividade

RSMSF é uma tecnologia recente que vem sendo alvo de vários estudos e tem sido bastante utilizada em diversos tipos de aplicações. Com o uso de recursos, como câmeras e microfones, é possível explorar aplicações que não eram viáveis através de sensores escalares. Dessa forma, uma RSSF passa a compor um sistema de visão distribuído, permitindo se obter dados de vários pontos de vista de um mesmo fenômeno. Com uma única câmera composta de um sistema de pan-tilt-zoom, por exemplo, onde há o controle da inclinação, giro panorâmico e zoom, a aplicação se limita apenas ao campo de visão alcançado em decorrência do uso desse recurso. Entretanto, com o uso de múltiplas câmeras de baixo custo, é possível aumentar a cobertura da aplicação além de torná-la mais robusta. Além disso, o uso dessas várias câmeras provê a redundância necessária para prevenção de falhas relacionadas a obstruções em uma determinada linha de visão.

Os nós sensores que compõem as RSMSF apresentam um consumo de energia mais elevado que os nós em RSSF tradicionais, pois operam com captura e processamento de imagem e áu-

dio, que são bastante custosos. Devido a esse fator, os projetos de sistemas de RSSF atuais geralmente são implementados de acordo com a Figura 1.7, ou seja, utilizando redes mistas e dispostas de forma hierárquica, onde os nós equipados com sensores multimídia são utilizados em menores quantidades e em lugares estratégicos, enquanto os nós mais simples são utilizados em grandes quantidades e distribuídos de forma mais abrangente. Esses dispositivos mais simples geralmente são responsáveis pela conectividade da rede por ser uma operação que não exige um alto poder de processamento e armazenamento. Já os dispositivos com suporte à recursos multimídia são capazes de executar tarefas mais complexas, além de oferecerem recursos adicionais (Hill et al. 2004).

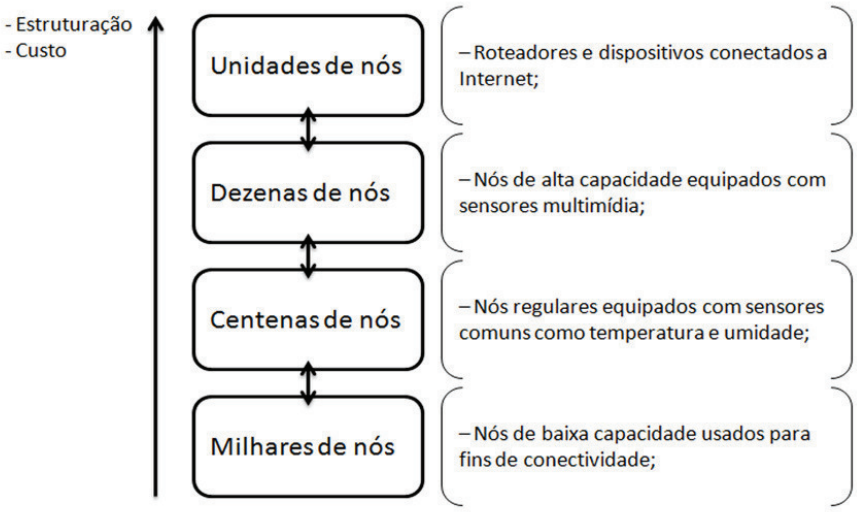


Figura 1.7: Arquitetura típica para uma RSMSF.

Existem várias implementações de RSMSF em cenários reais. Em (Akyildiz & Vuran 2010) duas aplicações são descritas, sendo que ambas exploram as capacidades das câmeras acopladas aos nós sensores. A primeira delas é o sistema SensEye. Ele é utilizado em aplicações de vigilância e tem o objetivo de realizar tarefas de detecção de objeto, reconhecimento e *tracking*. A arquitetura da rede segue um modelo multicamadas onde componentes heterogêneos com diferentes sensores e capacidades de processamento são dispostos em uma estrutura hierárquica. A camada mais baixa da rede consiste de dispositivos de baixa capacidade compostos por sensores de vibração responsáveis por detectar atividades no ambiente. Qualquer atividade percebida é

informada para a camada superiores para que seja feito o reconhecimento do objeto por câmeras de baixa resolução. A terceira camada, consiste de câmeras de média resolução conectadas a plataformas Stargate da antiga Crossbow, agora Memsic. Essas placas são capazes de se comunicar com todas as camadas inferiores além de prover um centro de controle através de uma conexão WiFi. A alta capacidade de processamento e armazenamento desses nós também são exploradas para executar funcionalidades de *gateways* assim como processamento de imagens de alto custo. Adicionalmente, a arquitetura apresenta uma quarta camada composta de câmeras pan-tilt-zoom de alta resolução conectadas a PCs para que processamentos mais complexos possam ser executados. O modelo hierárquico melhora a eficiência energética nesse cenário devido a atribuição de tarefas mais complexas à dispositivos mais poderosos, enquanto os mais simples são responsáveis por prover cobertura ao sistema.

O segundo sistema se trata do IrisNet. Ele é uma plataforma de *software* para desenvolvimento de serviços sobre RSMSF e permite que a rede de sensores possa ser usada através de consultas pela Internet. Sensores de vídeo e sensores escalares são espalhados pelo ambiente para realizar a coleta de dados relevantes. A rede permite o acesso a informações dos sensores de vídeo através da Internet. O usuário tem a rede como uma unidade abstrata que pode ser acessada através de linguagens de alto nível. A cada consulta é feita a coleta de dados pela rede de sensores, que permite desde consultas simples até consultas mais complexas envolvendo operadores aritméticos e banco de dados. A arquitetura é dividida em duas camadas. Sensores heterogêneos implementam uma interface compartilhada chamada de agentes de sensoriamento. Os dados produzidos pelos sensores são armazenados em um banco de dados distribuídos que é implementado nos agentes organizadores. Vários serviços de sensoriamento são oferecidos simultaneamente pela mesma arquitetura. Os dados de sensores são representados através de XML, o que permite fácil manipulação dos dados. Os agentes organizadores são responsáveis pelo serviço de sensoriamento, coleta de dados produzidos pelo serviço e organização das informações em um banco de dados distribuído, tornando as informações disponíveis para atender às consultas.

Com o uso de serviços, como os descritos, tecnologias complementares passaram a ser explo-

radas, como a robótica móvel por exemplo, que pode ser integrada à sistemas de RSSF (Souza 2011). Dessa forma, tornou-se viável o estabelecimento de ambientes assistivos para auxílio de pessoas que utilizam cadeiras de rodas motorizadas, de modo que possam realizar tarefas cotidianas independentemente. Nesse tipo de aplicação, através do sensoriamento, o ambiente é capaz de obter informações que podem ser úteis para estimar o mapa do local, a localização do alvo e até mesmo obstáculos que possam vir a comprometer o tráfego pelo ambiente. Os desafios para estabelecimento de um ambiente assistivo estão relacionados à sua arquitetura, funcionalidade, usabilidade, privacidade e custo. Estes desafios estão relacionados às restrições do usuário que demandam soluções intuitivas, não intrusivas e com um custo acessível. Além disso, é preciso que haja a cobertura necessária para o monitoramento contínuo do alvo (Vangelis Metsis & Makedon 2008). Diversos trabalhos na literatura abordam aplicações integrando as tecnologias citadas. Em (Guanling Chen & Wang 2008) é proposto um sistema de monitoramento baseado em câmeras para ambientes assistivos. Esse sistema realiza a localização de um alvo, como um paciente por exemplo, e propõe um algoritmo de seleção de câmera a fim de proporcionar uma cobertura ótima ao sistema e garantir o rastreamento contínuo do alvo. No cenário apresentado, o paciente é equipado com sensores corporais e diversos roteadores IEEE 802.11 são instalados de modo estratégico pelo ambiente. Cada roteador possui uma câmera IP e um nó sensor anexados, sendo que esse nó opera como um *gateway* para a RSSF (IEEE 802.15.4). A localização do paciente é baseada na força do sinal trocado periodicamente na forma de um *beacon*, entre o nó anexo ao roteador e aquele anexo ao próprio paciente. Após a localização ser estimada, o sistema elege as câmeras mais prováveis a cobrir o alvo. As câmeras selecionadas realizam a captura da imagem e caso detectem movimento, através da subtração do plano de fundo, a imagem é segmentada e transferida através da RSSF para um servidor, possibilitando um monitoramento contínuo. As principais restrições desse sistema é o uso de equipamentos de alto custo, o que inviabiliza a implementação do sistema em cenário real e a transmissão de imagens de forma não confiável utilizando a RSSF. Essa transmissão, além de se dar em baixas taxas, não garante a entrega da informação, além de impor uma grande sobrecarga ao sistema resultando em um desempenho reduzido.

Em (Kosmopoulos 2010) é apresentado um sistema para ambientes assistivos cujo objetivo é monitorar o comportamento de um paciente em tempo real. O sistema proposto utiliza visão computacional e estima o comportamento do paciente com base em dois critérios: trajetória do alvo e atividades em curto prazo. A trajetória e a localização do alvo são estimadas de modo que seja possível saber onde o paciente se encontra no momento de execução de uma determinada atividade. Para isso, o autor utilizou técnicas de subtração de plano de fundo e de rastreamento de marca. Já a estimação das atividades de curto prazo é feita com base na técnica de fluxo ótico. Através desse critério é possível saber se o paciente está executando algum movimento brusco ou até mesmo se está sentando, levantando, caminhando, etc. Apesar do sistema proposto apresentar resultados satisfatórios, ele faz uso de classificadores, o que exige um treinamento inicial para que dados obtidos através das imagens capturadas sejam transformados em informações relevantes ao sistema.

Em (Olsen & Hoover 1999), é apresentado um sistema de detecção de obstáculos baseado em RSMSF para auxilio na navegação de robôs móveis. O sistema possui um serviço de detecção que gera um mapa de ocupação do ambiente monitorado. Serviços de navegação vinculados a robôs podem receber estes mapas gerados com informações da rede para traçar rotas para os alvos desejados. O sistema gera mapas na mesma taxa de aquisição da câmera, desse modo os obstáculos dinâmicos são vistos pelo sistema como diferentes obstáculos estáticos para cada mapa. O mesmo ocorre para múltiplos robôs, onde robôs no ambiente são vistos como obstáculos entre si. O ponto negativo desta solução é que as informações de todas as câmeras são transmitidas a um servidor externo para processamento, desta forma, além da RSMSF não realizar processamento distribuído da informação, ainda é necessário que todas as imagens de todas as câmeras sejam transmitidas para este servidor, tarefa esta que é custosa e demanda um consumo elevado de energia nos nós sensores.

Em (Rekleitis, Meger & Dudek 2006) é apresentado um sistema que busca realizar a localização de um robô móvel através de uma RSMSF, enquanto simultaneamente o robô é utilizado para estimar a posição dos nós da RSMSF. Neste cenário o robô foi equipado com uma marca conhecida, detectada utilizando o sistema. Quando um nó sensor detecta a marca, ele troca

informações com o robô podendo assim estimar sua localização relativa ao mesmo. Para compensar o acúmulo do erro de odometria, o robô pode voltar a áreas previamente mapeadas e ajustar sua posição. Diferentes técnicas foram testadas sobre a melhor estratégia de movimentação do robô de forma a minimizar o erro das estimações e a distância percorrida. Já em (Meger, Marinakis, Rekleitis & Dudek 2009) os autores apresentam um método de localização distribuída onde apenas robô e cada nó sensor individualmente participam do processo de localização dos sensores.

O trabalho apresentado em (Mehta, Sheng, Chen & Shi 2009) descreve um sistema para realizar o rastreamento de um alvo móvel em um ambiente monitorado, cenário este, similar a outros já citados. O sistema apresentado utiliza informações de localização de um alvo que está sendo rastreado para estimar a posição, coordenadas cartesianas e orientação dos nós sensores distribuídos pelo ambiente. Neste caso o alvo deve estar equipado de acordo com alguma técnica de localização. A localização dos nós se dá de maneira cooperativa e distribuída. Dado a movimentação do alvo entre dois pontos, os sensores capazes de monitorar o movimento irão inferir e trocar informações entre si para determinar as novas estimativas de posicionamento. A escolha do sistema de localização pode se tornar um problema em sistemas de larga escala devido ao acúmulo de erros inerente a maioria das técnicas.

Uma abordagem similar para o problema de rastreamento em RSMSF é apresentado em (Liu, Zhang & Ma 2010). O processo de localização é dividido em duas fases, onde a primeira consiste essencialmente na escolha de quais dispositivos capazes de detectar o alvo irão participar do processo de localização. Em seguida os nós selecionados realizam o processo de localização do alvo. Quando um novo sensor passa a detectar o alvo e torna-se um candidato para realizar a localização o processo é repetido.

Mais uma abordagem para localização distribuída onde ocorre uma seleção sobre quais nós serão utilizados é apresentado em (Lin, Zeng, Jiang & Jin 2011). Porém neste caso é realizada uma localização do alvo em três dimensões. Nós que detectam o alvo enviam suas informações para uma base central onde um algoritmo baseado nos mínimos quadrados é encarregado de realizar a seleção dos nós que irão participar do processo de localização. Devido a localização

ser tridimensional existe a necessidade de no mínimo 3 ou 4 nós, dependendo do cenário, para obter-se resultados satisfatórios.

Um processo alternativo para localização de nós sensores, chamado LISTEN, é apresentado em (He, Shen, Liu, Mo & Dai 2010). Neste caso, a localização é realizada de maneira reativa sem a necessidade de grande troca de dados entre nós da rede. Um nó móvel cruza a área monitorada pela RSMSF enviando mensagens (*beacons*) em determinados pontos de sua rota. Os nós da rede ao receberem esta mensagem capturam imagens do ambiente e tentam detectar a marca do nó móvel para realizar a estimação da sua posição. Nesta solução os nós da rede apenas aguardam mensagens enviadas pelo nó móvel e jamais enviam respostas, o que diminui a quantidade de informações enviadas e o consumo de energia. Por outro lado, se faz necessário que o nó móvel realize um traçado de forma a maximizar a área percorrida e envie mensagens periodicamente ao longo de todo percurso, fazendo com que o volume de informação transmitida seja elevado e, consequentemente, o consumo de energia.

Esse capítulo abordou o estado da arte das RSSF através da descrição dos conceitos, terminologias, arquiteturas, sistemas operacionais, técnicas de comunicação e protocolos. Além disso, foi feita a contextualização dessa tecnologia no cenário atual e um levantamento de como podem ser aplicadas a cenários futuros de forma a atender as novas tendências. Finalmente, abordou-se o conceito de RSMSF e como essa subárea de RSSF vem afetando o desenvolvimento de aplicações, tornando-as mais inteligentes e sofisticadas. O capítulo seguinte irá apresentar a metodologia utilizada para desenvolvimento desse trabalho bem como os equipamentos utilizados, técnicas e procedimentos.

Capítulo 2

Infraestrutura de Rede

No capítulo anterior foi explorado os principais conceitos da área de RSSF e como elas podem ser aplicadas para estabelecer ambientes inteligentes. Além disso, foram descritos os protocolos, padrões e sistemas operacionais mais comuns aplicados à essa tecnologia, assim como o contexto e as tendências das principais aplicações que fazem seu uso. Nesse capítulo serão abordados os dispositivos, as ferramentas e os procedimentos utilizados para estabelecimento da infraestrutura de rede implementada no desenvolvimento desse trabalho.

2.1 Plataforma

A infraestrutura de rede foi estabelecida com base na plataforma *high-end* para RSSF da Memsic, o Imote2, que foi projetado para atender aplicações que demandam alto processamento e confiabilidade de transmissão (Crossbow 2007). A plataforma possui um processador Intel XScale PXA271 de 32 bits e 13-416 MHz, que é uma implementação da arquitetura ARMv5TE e inclui todo seu conjunto de instruções. Essa tecnologia de processadores é baseada na arquitetura RISC (*Reduced Instructions Set Computer*) e sua filosofia favorece o uso de instruções simples porém poderosas, capazes de serem executadas em um único ciclo em *clocks* de alta velocidade. Diferente da arquitetura CISC (*Complex Instruction Set Computer*), que é a base das arquiteturas dos computadores modernos, o RISC foca na redução da complexidade das instruções executadas pelo *hardware*, o que aumenta a complexidade no processo de compilação,

Figura 2.1.

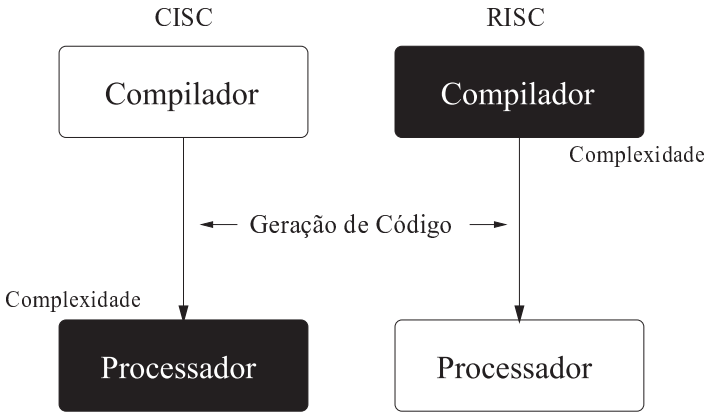


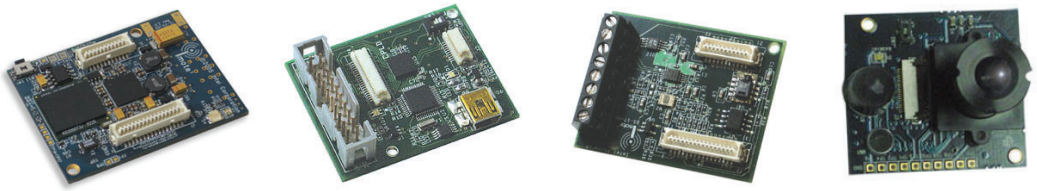
Figura 2.1: RISC x CISC.

O Imote2, Figura 2.2(a) também conta com a tecnologia Intel WMMX que atua como coprocessador, o que possibilita o aumento de desempenho no processamento de mídias, através do uso de instruções específicas capazes de otimizar processamentos comuns envolvendo esse tipo de dado. A plataforma integra um rádio 802.15.4 (TI CC2420) e uma antena *on-board* (2.4 GHz), possui memória SDRAM e *flash* de 32MB, além de disponibilizar vários componentes de E/S para conexão de dispositivos adicionais, como sensores e câmeras.

Para execução de funções de sensoriamento, foi utilizado o módulo desenvolvido para o próprio Imote2, o ITS400, Figura 2.2(c). Esse módulo possui quatro sensores escalares: luminosidade, temperatura, umidade e acelerômetro de 3 eixos. Além disso, a Memsic também oferece um módulo multimídia IMB2400 (Figura 2.2(d)) para o Imote2, contendo câmera, microfone e sensor de presença. Este último, particularmente, foi o módulo utilizado neste trabalho, contudo, apenas as funcionalidades da câmera foram exploradas. Trata-se de uma câmera OmniVision OV7670 colorida e com resolução de 640x480 pixels e taxa de captura que alcança até 30 quadros por segundo.

Outro módulo bastante importante é o IIB2400, Figura 2.2(b). Ele oferece uma interface para programação e troca de mensagens entre o dispositivo da rede e um computador pessoal. A comunicação física com a interface é feita através de uma porta USB, que é mapeada como duas UARTs através de um *driver* FTDI (FTDI 2012), sendo que uma é utilizada para a troca de men-

sagens e a outra para depuração. O módulo de programação acompanha uma porta JTAG, que permite descarregar programas no dispositivo através do *software* OpenOCD (OpenOCD 2012), incluindo sistemas operacionais. A plataforma Imote2 oferece suporte a diferentes sistemas operacionais. Seu sistema padrão é o TinyOS, entretanto, é possível utilizar versões simplificadas do Linux e do .NET *framework*.



(a) (b) (c) (d)

Figura 2.2: *Kit* da Memsic: (a) Imote2, (b) IIB2400, (c) ITS400, (d) IMB2400.

2.2 Sistema Operacional

Como o trabalho se baseia no estabelecimento de uma rede de sensores de multimídia, o que é favorecido pelo uso de dispositivos *high-end*, o sistema operacional escolhido foi o Linux. Diferentemente do TinyOS, com o uso do Linux é possível desenvolver aplicações mais complexas utilizando a linguagem C/C++. Além disso, se trata de um sistema robusto, bem conhecido e com uma vasta documentação, o que facilita o uso e a extensão de ferramentas e bibliotecas para a manipulação de imagens e vídeos.

A instalação do sistema operacional embarcado é feita através de três componentes: o *Bootloader*, o *Kernel* e o Sistema de Arquivos. No trabalho em questão optou-se por utilizar o Blob como *Bootloader*, a versão 2.6.29 do *Kernel* do Linux e um Sistema de Arquivos do tipo JFFS2 (*Journalling Flash File System version 2*). Essa versão do *Kernel* foi escolhida por oferecer suporte ao *hardware* do Imote2 e foi customizada para incluir apenas componentes essenciais, como *drivers* dos sensores e do rádio, além de ferramentas para execução de tarefas básicas como gerenciamento de arquivos, acesso remoto, entre outras. Os componentes do sistema operacional ficam dispostos na memória de acordo com a Figura 2.3. Para realizar a compilação desses

componentes foi utilizado um *toolchain* para ARM contendo o compilador GCC versão 4.1.2 e diversas bibliotecas de C/C++. Como se trata de um sistema embarcado, essa compilação deve ser feita através de compilação cruzada.

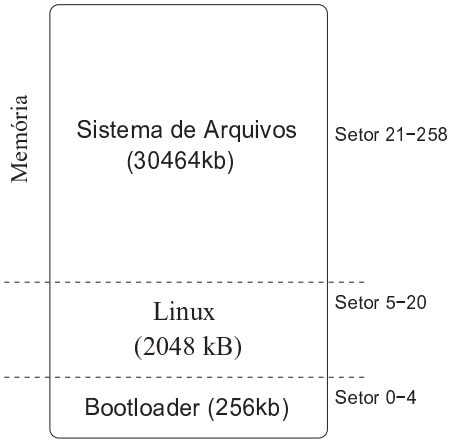


Figura 2.3: Instalação do sistema Linux na memória do Imote2.

A compilação cruzada é necessária quando a máquina onde se faz a compilação de um código-fonte não possui a mesma arquitetura comparada a máquina onde o código irá ser executado. No caso em questão os códigos-fonte foram gerados para arquitetura ARM a partir de uma arquitetura x86 e em seguida os binários gerados foram descarregados no dispositivo utilizando o módulo de programação através de um cabo JTAG e o OpenOCD. O OpenOCD basicamente estabelece uma conexão entre o dispositivo de *hardware* e o PC, oferecendo um serviço de depuração para sistemas embarcados. Dessa forma, é possível se conectar ao serviço através de uma conexão Telnet, por exemplo, e ler informações referentes ao estado do *hardware*, assim como enviar comandos de escrita em memória. A Figura 2.4 apresenta o cenário necessário para a instalação do Linux no Imote2.

O processo de compilação e instalação do sistema operacional para o Imote2 não é bem documentado, por isso essa etapa do trabalho foi executada com base em fóruns de discussões referentes ao assunto. Adicionalmente, é possível encontrar compilações e *toolchains* específicos para a plataforma em repositórios na Internet utilizados pela comunidade de desenvolvimento em RSSF e sistemas embarcados.

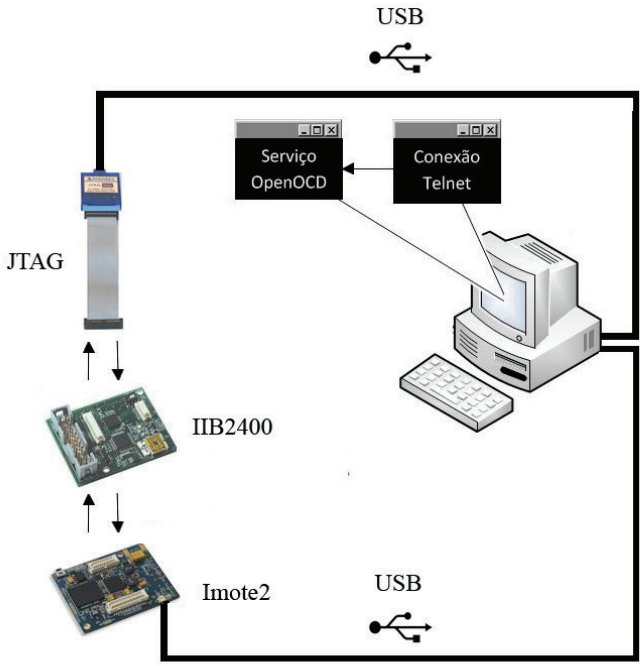


Figura 2.4: Cenário para instalação do Linux.

2.3 Comunicação

Para o sistema operacional escolhido, a plataforma Imote2 possui um protocolo de camada 2 chamado TOSMAC (Kasteleiner 2010). Esse protocolo atende as especificações do padrão IEEE 802.15.4 para realizar a comunicação sem fio. Apesar de oferecer um protocolo de camada 2, não existem protocolos de camada 3 desenvolvidos para a plataforma usando o sistema Linux. A falta de um mecanismo de roteamento limita a comunicação apenas aos vizinhos dentro da área de alcance do sinal. Para solucionar esse problema foi desenvolvida nesta dissertação uma versão simplificada do protocolo AODV (*Ad hoc On-Demand Distance Vector*) (Perkings 2003). O AODV foi escolhido principalmente pela sua simplicidade e eficiência em cenários controlados de pequeno porte, e por ter várias implementações bem sucedidas em cenários reais. Ele foi desenvolvido para redes *Ad Hoc* baseadas no protocolo IP, porém a implementação desenvolvida utilizou identificadores de camada 2 para endereçar os nós. Além disso, mecanismos mais complexos como respostas de rota a partir de nós intermediários e manutenção automática de rotas foram omitidos.

A operação do protocolo é baseada em três tipos de mensagens: RREQ (*Route Request*), RREP (*Route Reply*) e RERR (*Route Error*). Cada nó que deseja enviar um pacote de dados para um determinado destino envia um RREQ em difusão para seus vizinhos. Se o nó que recebeu a mensagem em difusão do originador da RREQ não for o destino definido na mensagem, o nó corrente cria uma rota para o nó que originou a difusão e encaminha a mensagem para os seus vizinhos. Esse processo ocorre sucessivamente até que o destino definido na mensagem de solicitação de rota seja alcançado. O protocolo AODV utiliza um mecanismo de prevenção de laços baseado no identificador de RREQ, ou seja, cada nó possui um registro das requisições que recebeu e o nó que a originou. Caso essa requisição já tenha sido recebida anteriormente ela é descartada sem enviar qualquer tipo de notificação. Quando um RREQ alcança um destino, é gerado um RREP que é enviado para a origem em *unicast* pela rota de menor custo encontrada pelo protocolo, que nesse caso é a que se encontra a menos saltos de distância da origem. A partir de então o nó que solicitou a rota é capaz de encaminhar mensagens para aquele determinado destino. O protocolo AODV também é capaz de realizar a manutenção de rota, ou seja, quando um nó detecta a perda de um nó vizinho que faz parte da rota até um destino, ele notifica seus outros vizinhos que aquele nó não está mais presente na topologia. Essa verificação é feita através do envio frequente de mensagens HELLO.

Antes do protocolo ter sido implementado nos dispositivos reais, ele foi desenvolvido e testado em um simulador especifico para RSSF, o Castalia (Boulis 2011), para validação de seus mecanismos. O Castalia é um simulador baseado na plataforma OMNeT++ que oferece modelos bastante realísticos do canal de comunicação sem fio, do rádio, do comportamento dos nós e de outras características típicas desse tipo de rede. Esse simulador possui a interface bem semelhante à interface nativa do TOSSIM (Levis & Le 2003). Ele é baseado em linhas de comando e oferece modelagens dos rádios CC2420 e o CC1000, entretanto, diferentemente do TOSSIM, sua programação é em C/C++ e oferece implementações de protocolos de camada MAC, como o S-MAC, o T-MAC e o IEEE 802.15.4 MAC. O simulador foi desenvolvido pelo NICTA (*National Information and Communications Technology Research Centre of Australia*) e sua modelagem é baseada em medidas empíricas feitas em laboratório utilizando plataformas

específicas para RSSF, como o MicaZ.

O protocolo foi testado com dispositivos reais dispostos em diferentes topologias utilizando 4 nós. Com isso foi possível validar o funcionamento de alguns dos mecanismos implementados descritos na RFC, como inundação controlada de requisições de rota, geração de rota, encaminhamento de mensagens através de múltiplos saltos, notificação de rotas inválidas, etc. Em média, o tempo necessário para um nó processar uma mensagem e difundi-la pela rede é de aproximadamente 156ms. O código do protocolo possui 1424 linhas e ocupa 41.8Kb de memória. Evidentemente, com o aumento significativo do número de nós, o desempenho da rede deve diminuir consideravelmente afetando no seu processo de convergência e consequentemente comprometendo a escalabilidade. Porém, como esse trabalho é proposto para ambientes fechados e controlados, e como não tem foco na análise de desempenho do protocolo de roteamento, mas sim na aplicação, os fatores citados não são considerados críticos. A Tabela 2.1 lista as operações do AODV, extraídas de Perkings (2003), que foram implementadas neste trabalho.

Tabela 2.1: Operações implementadas do AODV.

Operação	Estado
Manutenção de Números de Sequência	Implementado Parcialmente
Entradas de Tabela de Rota e Listas Precursoras	Implementado Parcialmente
Geração de Requisições de Rota	Implementado
Controle de Disseminção de Requisições de Rota	Implementado
Processamento e Encaminhamento de Requisições de Rota	Implementado
Geração de Respostas de Rota	Implementado Parcialmente
Recebimento e Encaminhamento de Respostas de Rota	Implementado
Operações Sobre *Links* Unidirecionais	Não Implementado
Mensagens *Hello*	Implementado
Manutenção de Conectividade Local	Implementado Parcialmente
Mensagens de Erro, Expiração e Remoção de Rota	Implementado Parcialmente
Reparo de Rota Local	Não Implementado
Ações de Reinicialização	Não Implementado
Interfaces	Não Implementado

2.4 Processamento de Imagem

Para realizar o processamento embarcado de imagens, optou-se pela utilização da biblioteca OpenCV (Gregory 2012). O OpenCV é uma ferramenta de visão computacional e foi escolhido

por ser bem estabelecido, possuir uma comunidade ativa de desenvolvedores e por oferecer diversos filtros e métodos para o processamento de imagens e vídeos, favorecendo o desenvolvimento ágil. Além de seus benefícios, um fator crucial para a escolha do OpenCV foi a sua portabilidade para a arquitetura ARM, que foi utilizada no desenvolvimento deste trabalho. Adicionalmente, o OpenCV oferece suporte para linguagem C/C++, permitindo o desenvolvimento de aplicações de alta complexidade. Para portar a biblioteca para a plataforma ARM, foi utilizado o mesmo *toolchain* de geração dos arquivos binários equivalentes ao sistema operacional, processo esse já descrito anteriormente. Em seguida, o instalamos na plataforma via SSH (Secure Shell). O uso do SSH foi possível através de um *driver* Ethernet-*over*-USB (módulo usbnet do *kernel* do Linux), que é capaz de emular uma conexão Ethernet utilizando uma conexão serial (USB) como meio físico. Dessa forma foi possível atribuir endereçamento IP para os dispositivos, criando uma comunicação ponto-a-ponto entre eles. A Figura 2.5 mostra o funcionamento deste cenário.

O processador ARM por padrão não oferece suporte à aritmética de ponto flutuante via *hardware*. Duas soluções geralmente oferecidas são o uso de um acelerador de vetor de ponto flutuante implementado através de um coprocessador ou então pela emulação via software seguindo o padrão IEEE 754.1985. No nosso caso, a plataforma Imote2 oferece um co-processador iWMMXt implementado para tratar operações envolvendo imagens e vídeos. A escolha do método para operações de pontos flutuantes é definida durante a compilação através de diretrizes passadas para o compilador.

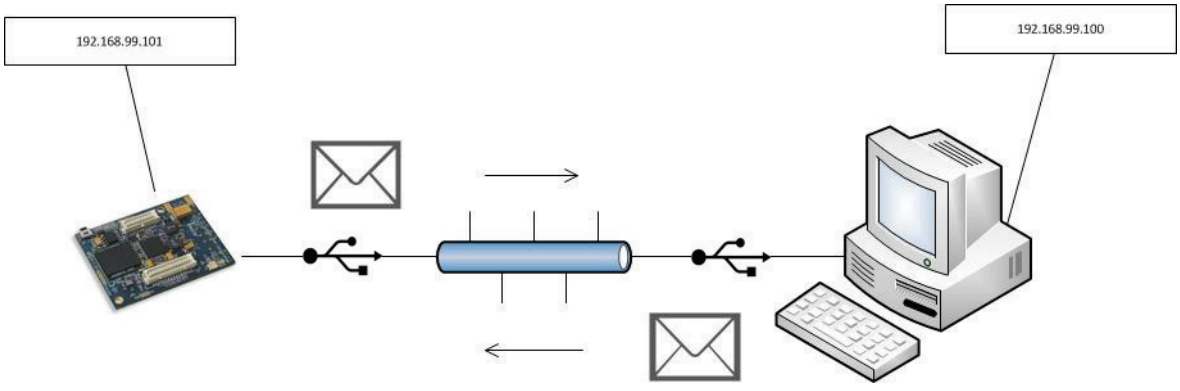
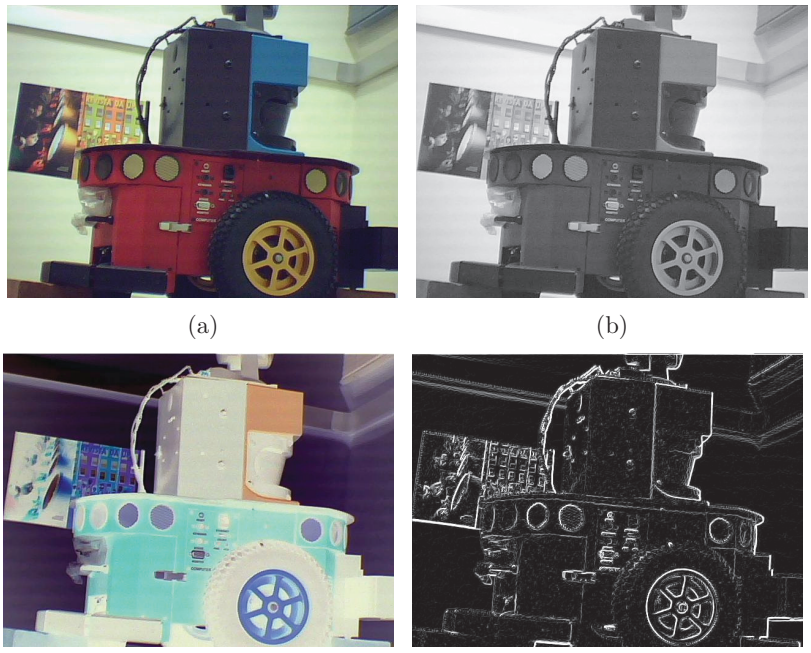


Figura 2.5: SSH utilizando Ethernet-*over*-USB.

A Figura 2.6 mostra uma imagem capturada e processada através do Imote2 equipado com

módulo de câmera, onde a Figura 2.6(a) é a imagem original e as demais são imagens processadas através da ferramenta OpenCV. Neste caso são testados funcionalidades básicas da ferramenta, como filtros e transformações. Na Figura 2.6(b) é feita uma conversão para escala de cinza, na Figura 2.6(c) é feita a extração do negativo da imagem e na Figura 2.6(d) é feita a detecção de bordas através do método de Sobel. O tempo de processamento das imagens citadas, desconsiderando a imagem capturada, foi de 70ms, 180ms e 360ms, respectivamente.



(a) (b)

(c) (d)

Figura 2.6: Imagens obtidas a partir da câmera do Imote2: (a) Imagem original, (b) Tons de cinza, (c) Negativo, (d) Detecção de bordas.

Nesse capítulo foi apresentada a infraestrutura necessária para o desenvolvimento e execução das aplicações robóticas desenvolvidas ao longo deste trabalho. A plataforma de RSSF utilizada, o Imote2, foi descrita e, suas características e aplicações abordadas. Em seguida comentou-se sobre a instalação do sistema operacional Linux no Imote2 e o desenvolvimento do algoritmo de roteamento que possibilita a troca de mensagens entre dispositivos afastados geograficamente. Por fim comentamos brevemente sobre o OpenCV, uma ferramenta de visão computacional que foi instalada em todos os dispositivos sensores permitindo o processamento de imagens associadas a diferentes pontos de vista no ambiente. O capítulo seguinte irá detalhar

os serviços propostos neste trabalho, explicando o funcionamento e a metodologia utilizada para o desenvolvimento de cada aplicação.

Capítulo 3

Serviços Oferecidos pela Rede

O capítulo anterior descreveu a infraestrutura necessária para que os serviços propostos neste trabalho fossem desenvolvidos. Esses serviços foram propostos com objetivo de auxiliar robôs móveis semi-autônomos na execução de suas tarefas. Eles visam contribuir com problemas clássicos da robótica móvel: a localização e a navegação.

O serviço de localização é importante para que um robô, durante sua navegação pelo ambiente, saiba onde se encontra em determinado instante de tempo e estime o quão próximo está de atingir seu objetivo. Dependendo do ambiente, vários obstáculos de diferentes naturezas podem surgir pelo seu caminho durante o percurso, impondo o uso de novas estratégias de geração de rotas para o cumprimento de sua meta.

Para a navegação, é preciso um mecanismo adequado para controle das rodas do robô e que seja capaz de guiá-lo. O objetivo do serviço de navegação é auxiliar na escolha da melhor rota de navegação de tal forma que o percurso seja adequado e favoreça o cumprimento do objetivo, de forma segura e eficiente. Esse capítulo irá descrever com detalhes os serviços desenvolvidos.

3.1 Serviços Desenvolvidos

Devido ao consumo de energia e a carga de processamento mais elevada em comparação as plataformas de RSSF *low-end*, resultantes do processamento de imagens e das características do sistema operacional escolhido, os serviços propostos nesse trabalho foram desenvolvidos para

serem aplicados em ambientes estruturados e controlados, onde as restrições em relação ao consumo de energia são menores. Nesse tipo de ambiente se espera que os nós possam ser alimentados continuamente e que a manutenção e gerenciamento da rede sejam facilitados. Além disso, os serviços consideram que o mapa do local e a posição dos nós sensores são conhecidos. Nas aplicações que irão ser descritas, um robô móvel é o nó Servedouro da RSSF, ou seja, ele que solicita os serviços oferecidos pelo ambiente.

Os serviços oferecidos foram baseados no processamento *in-networking*, ou seja, é realizado pelos próprios nós da RSSF. Outra opção seria transmitir as imagens para um computador dedicado, porém como o envio de dados é a tarefa que mais consome energia em um nó sensor e pelo fato da transmissão de imagens ser altamente custosa, esse método provocaria um aumento considerável da sobrecarga e consequentemente do tempo de resposta do sistema. Os serviços, basicamente, exploram a detecção de movimento e a localização de um alvo conhecido anexado ao robô. A detecção de movimento é feita através da comparação de leituras dos sensores extraídas em instantes de tempo distintos (domínio espaço-temporal). Já a localização pode ser feita através de *target tracking*, ou seja, pela extração de uma marca conhecida e estimação de sua distância em relação ao ponto focal da câmera.

As aplicações desenvolvidas requerem que o ambiente seja coberto pela infraestrutura de rede de modo que possibilite o monitoramento de todas as áreas navegáveis. No nosso caso, todos os nós reportam seu resultado sob demanda ao nó Servedouro, que se trata de um nó anexado a um robô móvel através de uma conexão USB. Dessa forma, os dados são recebidos pelo nó da RSSF e enviado ao robô, que por sua vez pode combiná-los e usá-los para criar estimativas relevantes.

3.1.1 Serviço de Navegação

O primeiro serviço definido foi o de auxílio à navegação de robôs móveis. Este serviço tem como objetivo auxiliar a navegação com base na taxa de ocupação do ambiente (técnica conhecida como *Crowd Detection*), a fim de prover informações ao robô sobre o quão ocupada está a rota até o seu destino. Nesse cenário vários nós equipados com câmeras são dispostos

pelo ambiente onde um ou mais nós se encarregam de monitorar um determinado corredor, de forma que todo o ambiente seja coberto. Assim, durante seu percurso, o robô pode solicitar ao ambiente informações referentes a um determinado corredor e então tomar decisões referentes à sua navegação de acordo com a estimativa retornada, como desviar do corredor caso esteja congestionado, diminuir a velocidade ou emitir um sinal sonoro, por exemplo. O cenário da aplicação é apresentado na Figura 3.1.

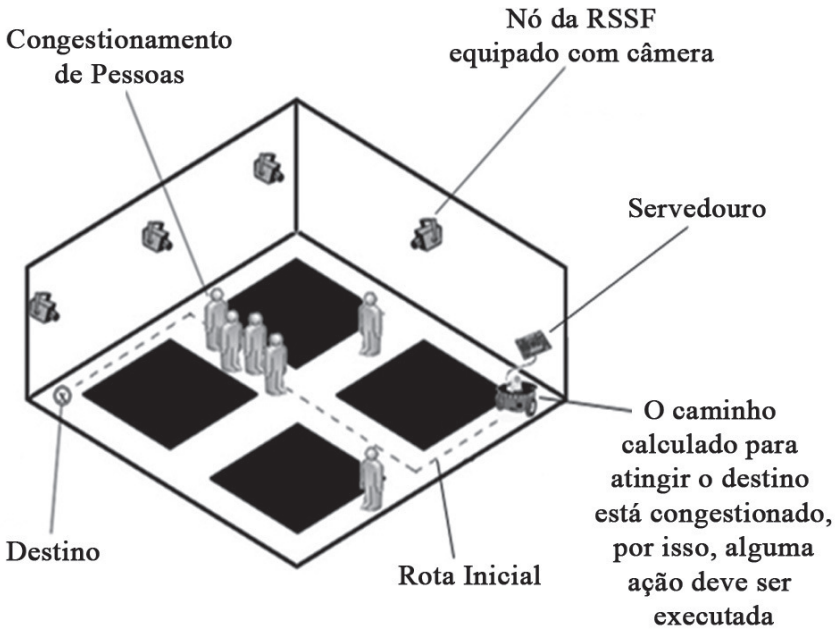


Figura 3.1: Cenário geral da aplicação de auxílio à navegação.

O valor retornado pela rede é um peso P entre 0% e 100%. Um ou mais nós da rede são responsáveis por monitorar um determinado corredor e gerar resultados associados a ele. Esses valores são enviados para o robô, que irá então atribuí-los aos corredores correspondentes definidos em seu mapa, possibilitando o cálculo da viabilidade de cada rota. Os nós da rede consultados são aqueles que estão instalados ao longo da rota inicial calculada pelo robô. Cada nó é consultado de acordo com a sua distância em relação ao robô, onde os mais próximos tem maior prioridade do que os outros. Desse modo, a resposta do sistema é mais condizente com o estado atual da respectiva área coberta devido a sua proximidade à posição do robô, o que

minimiza problemas relacionados a integridade da informação. O algoritmo de busca usado para planejamento de rota do sistema foi o D* (*Dynamic* A*), que é uma variação do algoritmo clássico A* (Stentz 1995). O algoritmo A*, basicamente, calcula a rota de menor custo entre duas posições do mapa por meio de uma heurística otimista denotada pela Equação 3.1, onde G é a distância percorrida até o ponto corrente e H é a distância estimada de forma otimista (por exemplo, a distância Euclidiana) até o ponto objetivo. A Figura 3.2 apresenta o mapa do ambiente, onde o robô deve analisar os custos para percorrer um caminho de uma determinada origem até um destino e selecionar aquele de menor custo.

$$F = G + H \tag{3.1}$$

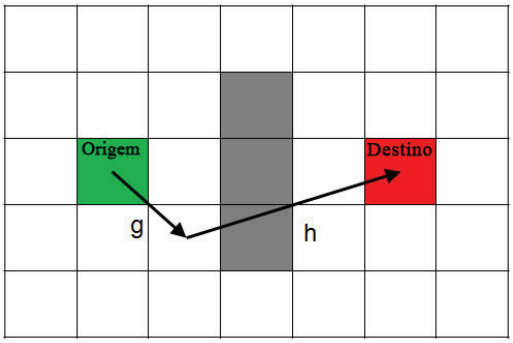


Figura 3.2: Cálculo do custo do caminho pelo algoritmo D*.

O mapa do ambiente é discretizado em células, onde cada uma delas é associada ao peso P de acordo com o custo para percorrê-la. Esse algoritmo de busca considera que o ambiente seja estático e de total conhecimento do robô. O calculo da rota é feito apenas uma vez antes do robô iniciar sua navegação e o caminho de menor peso é considerado o caminho ótimo. Nesse caso, quanto maior for o número de células utilizadas na discretização mapa, mais precisa é a navegação, porém o desempenho é afetado devido a maior carga de processamento necessária.

Ao contrário do algoritmo A*, o D*, apesar de utilizar a mesma lógica de busca, foi criado para lidar com ambientes dinâmicos. Ele considera que um caminho inicialmente viável pode se tornar inviável ao longo do percurso, o que indica que o robô não possui total conhecimento do ambiente. Por isso, além de calcular a rota mais curta até o destino, com o D*, o robô

realiza a leitura do ambiente através de seus sensores em intervalos de tempo definidos, obtendo assim informações pertinentes para sua navegação. Dessa forma, a cada iteração, a rota de menor custo calculada é aquela que além de ser mais curta, é a de menor peso considerando as informações obtidas.

No sistema proposto nesse trabalho, inicialmente todas as células do mapa são definidas com pesos mínimos ($P = 0$), indicando que os caminhos estão livres, com isso o robô inicialmente apenas realiza o cálculo da menor rota entre o ponto de início até o ponto de destino. Após o início da navegação, o robô passa a interagir com o ambiente em intervalos de tempo pré-definidos a fim de obter informações atualizadas a respeito do mapa. Dessa forma, o robô é capaz de atualizar o custo de cada rota do seu mapa e então recalcular a rota ótima a partir de sua posição corrente com base na sua nova configuração de mapa. O processo é repetido até que o robô alcance seu destino ou até que seja determinado que o destino não possui rotas navegáveis e não pode ser alcançado.

Para guiar as rodas do robô ao longo do trajeto foi utilizado um controlador *fuzzy* no modelo de MAMDANI com composição MAX-MIN e defuzzyficação por centroide. A entrada utilizada para o controlador foi o ângulo (α), expresso pela Equação 3.2, onde ΔX_H e ΔY_H compõe a distância do robô aos vários pontos na rota calculada pelo D*, Φ o ângulo dado pela odometria e $atan2$ é a função arco-tangente em radianos. A implementação do sistema de controle do robô é de autoria do aluno de doutorado Leonardo Rocha Olivi. Detalhes de implementação, assim como as técnicas e modelos citados são descritos em (Olivi, Souza, Paolieri, Guimaraes & Cardozo 2012).

$$\alpha = -\Phi + atan2\left(\Delta Y_H, \Delta X_H\right) \tag{3.2}$$

O algoritmo para estimar a taxa de ocupação do ambiente foi baseado na técnica de subtração de plano de fundo. As imagens são capturadas continuamente e, após serem convertidas para tons de cinza, são comparadas à uma imagem base, obtida previamente, referente ao plano de fundo inicial do ambiente sem ocupação. Essa comparação é feita através da subtração aritmética dos pixels das duas imagens comparadas, onde cada um deles é representado por

um valor entre 0 e 255 em uma escala de cinza. Após essa etapa, é feita a binarização na imagem resultante através da aplicação de um *threshold*, de modo a representar as mudanças ocorridas a partir da imagem base (plano de fundo). A cada iteração, uma nova captura é feita e a imagem base e a imagem corrente são comparadas, e isso ocorre continuamente enquanto o algoritmo estiver em execução. A Figura 3.3 ilustra as operações realizadas sobre as imagens na subtração de plano de fundo, onde a Figura 3.3(a) apresenta a imagem base do ambiente (tons de cinza), capturada previamente enquanto o mesmo esteve vazio. A Figura 3.3(b) mostra uma imagem capturada contendo variações de pixels, o que indica a presença de algum objeto na área de captura da câmera e a Figura 3.3(c) corresponde ao resultado da subtração modular entre a Figura 3.3(a) e Figura 3.3(b), pixel a pixel. Já a Figura 3.3(d) é a imagem binarizada após a aplicação de um *threshold* na imagem resultante (Figura 3.3(c)). Com isso, soma-se a quantidade total de pixels da imagem corrente que tiveram seus valores modificados em relação aos mesmos pixels da imagem base. Caso esse valor seja maior que aqueles calculados por outros nós sensores monitorando outros caminhos, então considera-se o caminho monitorado pelo nó sensor em questão menos viável para navegação.

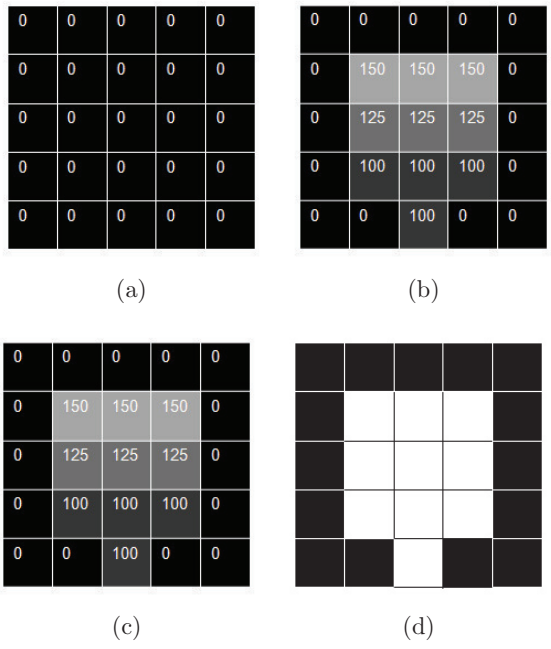


Figura 3.3: Subtração de plano de fundo: (a) Imagem base, (b) Imagem capturada, (c) Imagem resultante após subtração de pixels, (d) Imagem binarizada.

3.1.2 Serviço de Localização

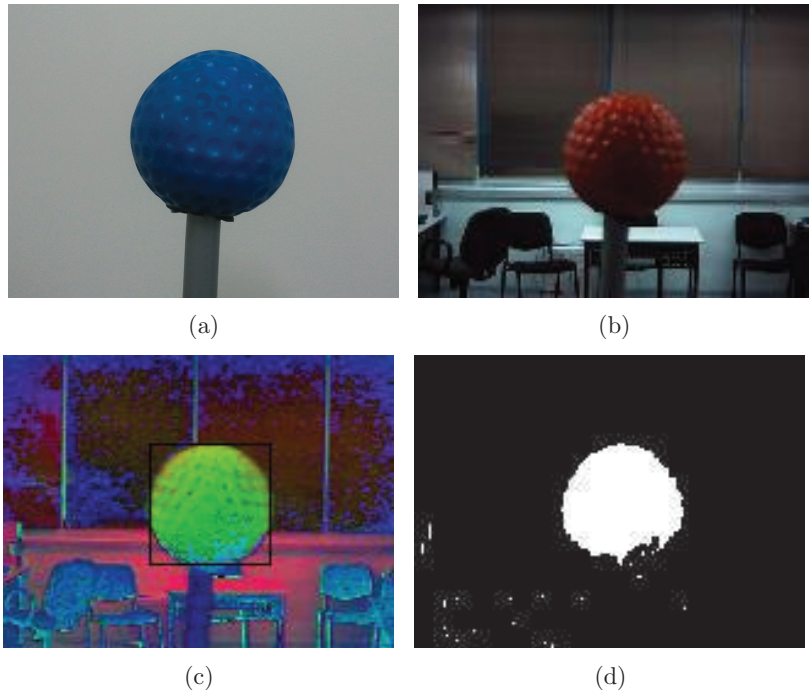
O segundo serviço implementado foi o de localização, que teve seu desenvolvimento baseado na detecção de uma marca anexada ao robô móvel. No cenário em questão, utilizou-se uma esfera de cor azul suspensa por uma haste sobre a base do robô, Figura 3.4.

Figura 3.4: Robô Seekur Jr. com haste anexada.

Nessa aplicação, o robô pode solicitar a qualquer momento ao ambiente uma estimativa da sua localização, que pode ser utilizada tanto para correção de odometria como para conhecimento de sua posição topológica. Ao receber a solicitação de localização, os nós vizinhos do robô realizam a captura de uma imagem e, caso detectem a marca (esfera de cor azul), iniciam os cálculos referentes a estimativa da posição da marca. Nesse cenário, considera-se que a esfera azul está suspensa a uma altura fixa H_B e que os nós espalhados pelo ambiente estão instalados a uma altura fixa H_M, onde $H_B = H_M$. Embora a marca utilizada seja de cor azul, a captura feita ocorre utilizando o sistema de cores BGR (*Blue Green Red*) como formato de pixel, ao invés de RGB (*Red Green Blue*). Por isso, a cor de interesse nas imagens capturadas nesta aplicação é o vermelho.

A detecção da marca é feita através da conversão da imagem capturada para o sistema de cores HSV (*Hue Saturation Value*). Após essa conversão, as componentes *hue*, *saturation* e *value* são separadas e em seguida submetidas a um *threshold*, de modo a isolar na imagem apenas a cor da marca utilizada. Após a extração da cor desejada, aplica-se a binarização da

imagem, convertendo o objeto de interesse para cor branca e o conteúdo sem relevância para preto. A Figura 3.5 apresenta uma imagem da marca capturada por um nó sensor e o resultado dos tratamentos feitos até o seu reconhecimento. A Figura 3.5(a) apresenta a marca original, a Figura 3.5(b) apresenta a imagem em um sistema de cores BGR, a Figura 3.5(c) apresenta o processamento feito para obtenção dos componentes HSV e detecção da marca desejada, e a Figura 3.5(d) apresenta o resultado da binarização, onde a parte branca corresponde à região de interesse da imagem capturada. Alguns ruídos também podem ser observados nesta imagem.



(a) (b)

(c) (d)

Figura 3.5: Resultados da extração e detecção da marca: (a) Marca original. (b) Imagem da marca em sistema de cores BGR, (c) Marca detectada, (d) Marca extraída.

Com a detecção da marca, calcula-se a centróide e seu diâmetro em pixels. Dessa forma é possível saber o valor estimado de sua profundidade em relação a posição da câmera. A atribuição de distância métrica em relação ao diâmetro em píxeis foi realizada através da calibração do algoritmo, que gerou uma função potencial, Figura 3.6.

A função gerada é definida pela Equação 3.3, onde Z é a distância métrica que a esfera se encontra da câmera em centímetros e D o diâmetro da marca na imagem, em pixels.

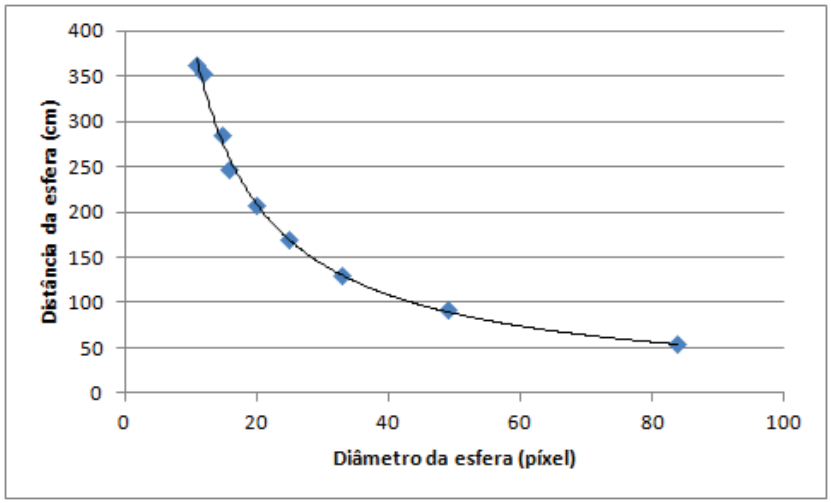


Figura 3.6: Função potencial que representa a relação entre o diâmetro e a distância métrica da esfera.

$$Z = 3577,3D^{-0.948} \tag{3.3}$$

Como o ângulo de abertura das câmeras utilizadas é muito pequeno, ele foi desprezado de acordo com dados empíricos. Comparando a Figura 3.7(a) e Figura 3.7(b), é possível verificar que a esfera apresenta tamanhos semelhantes em ambos os casos, mesmo após o deslocamento realizado. Dessa forma, considerou-se que a distância Z da esfera em relação a câmera na Figura 3.7(a) é aproximadamente igual a distância Z da esfera em relação a câmera na Figura 3.7(b).

(a) (b)

Figura 3.7: Imagens obtidas a partir duas posições distintas: (a) Esfera posicionada no centro do ângulo de abertura da câmera, (b) Esfera posicionada no limite do ângulo de abertura da câmera.

O valor do eixo X da posição do robô no ambiente foi associado ao eixo X referente a posição da centróide da esfera na imagem capturada, pois verificou-se que são empiricamente similares. A Figura 3.8 mostra como os eixos globais do ambiente são representados na imagem da marca capturada, onde Z é a profundidade, ou seja, a câmera é posicionada nessa direção, X é a posição da centroide da marca na imagem capturada e Y é a altura da marca, que sempre será fixa. Os valores referentes à posição do robô são convertidos para centímetros e em seguida é aplicada uma matriz de transformação, dada pela Equação 3.4 (Siegwat and Nourbakhsh, 2004), onde ΔZ e ΔX são as posições globais do nó sensor em um plano bidirecional, de acordo com um plano de referência definido, e Θ é o ângulo em que o nó sensor é posicionado, considerando que este sempre estará a 90 graus em relação ao chão.

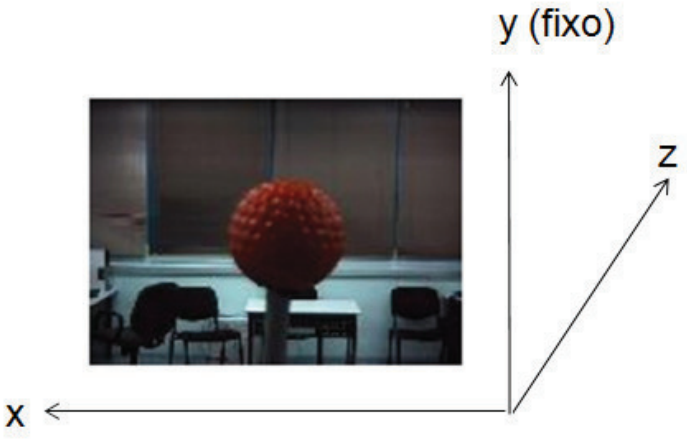


Figura 3.8: Representação dos eixos globais do ambiente na imagem da marca.

$$\begin{aligned} Z_1 &= \Delta Z + Z_0 \cos\Theta - X_0 \sin\Theta \\ X_1 &= \Delta X + X_0 \cos\Theta - Z_0 \sin\Theta \end{aligned} \tag{3.4}$$

Nesse serviço é necessário que o robô esteja parado para que sua localização seja calculada, pois no caso de estar em movimento seria necessário que houvesse a estimação do seu deslocamento durante o tempo gasto no processamento da marca. Apesar de não ser abordado neste trabalho, isso poderia ser alcançado através da obtenção do instante de captura e do instante

do fim do processamento. Como a velocidade média do robô é conhecida, bastaria calcular o deslocamento através da Equação 3.5. Para viabilizar essa solução, seria necessário que o intervalo entre as capturas fosse mínimo e que a orientação do robô fosse conhecida. Ainda assim, esse método poderia falhar em casos que o robô mudasse sua orientação durante o momento de processamento da marca.

$$\Delta S = \Delta T \cdot Vm \tag{3.5}$$

Esse capítulo apresentou uma descrição dos serviços propostos neste trabalho, assim como suas lógicas de funcionamento e conceitos necessários para implementá-los. No capítulo seguinte serão apresentados os resultados referentes aos estudos de caso aplicados para testar o comportamento dos serviços e verificar desempenho e precisão de resultados.

Capítulo 4

Experimentos e Resultados

O capítulo anterior apresentou os serviços desenvolvidos neste trabalho, descrevendo suas principais características, parâmetros e métricas utilizadas. Esse capítulo descreve os experimentos realizados para validação desses serviços e os resultados obtidos durante o trabalho. Todos os experimentos foram realizados em cenários reais a fim de se analisar de forma prática e consistente o comportamento das aplicações desenvolvidas. A avaliação das aplicações foram feitas com base no tempo de resposta de cada sistema e no erro de estimação.

4.1 Navegação

Antes de ser testado em ambiente real, esta aplicação foi executada em um ambiente simulado, empregando-se o simulador MobileSim (MobileSim 2011), utilizando dados extraídos do ambiente real. A simulação foi utilizada para análise dos recálculos de rota e das decisões que deveriam ser tomadas pelo robô. Além disso, possibilitou a definição da velocidade máxima aceitável durante o percurso até que o destino fosse alcançado. A definição desse parâmetro foi necessária para que não houvesse comprometimento ao sistema, pois como há limitações relacionadas às baixas capacidades de processamento dos dispositivos utilizados, uma velocidade elevada resultaria na inconsistência entre o corredor monitorado e o corredor ocupado pelo robô em um dado instante de tempo. No ambiente virtual foi gerado um mapa condizente com o ambiente real e os pesos referentes a cada corredor puderam ser atribuídos a partir das leituras

dos dispositivos (nós sensores) reais instalados em laboratório. Nesse experimento foram utilizados apenas três nós para monitorar um corredor real, porém, devido a grande quantidade de corredores a serem monitorados no ambiente virtual, as informações obtidas a partir destes dispositivos foram atribuídas a esses corredores de forma aleatória, isto porque um dos requisitos do serviço de navegação, desenvolvido neste trabalho, é a total cobertura do ambiente por nós sensores. Assim, foi possível visualizar as variações de trajeto do robô ao longo de todo o percurso, até que o destino fosse alcançado.

A Figura 4.1 apresenta o resultado da subtração de plano de fundo feita a partir do monitoramento de um corredor real utilizado no experimento. Este algoritmo foi implementado com 344 linhas de código e ocupa 8.1kB de memória do nó sensor.

Figura 4.1: Resultado do monitoramento de um corredor para diferentes ocupações.

O recálculo feito pelo algoritmo D* se deu com base nos pesos atribuídos aos corredores do ambiente, dado o conhecimento prévio do mapa do local. Aquele caminho de menor custo dentre todos os possíveis, passa a ser o escolhido para ser navegado. A Figura 4.2 mostra os resultados da navegação do robô no ambiente simulado. Comparando a rota inicial determinada pelo algoritmo D*, Figura 4.2(a), com o traçado final percorrido pelo robô, Figura 4.2(d), é possível observar dois momentos em que o caminho a ser percorrido esteve congestionado devido

a uma alta taxa de movimentação, forçando o robô a fazer o recálculo da rota, Figura 4.2(b) e Figura 4.2(c).

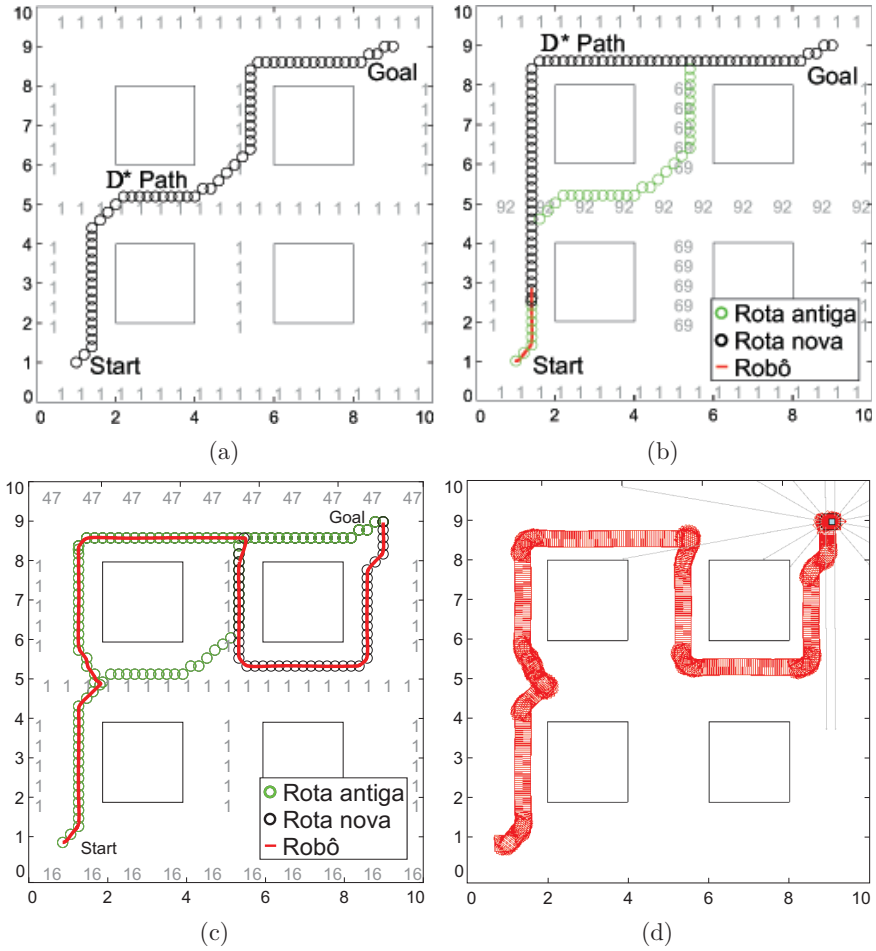


Figura 4.2: Resultados da simulação de recálculo de rota: (a) Rota inicial gerada, (b) Recalculo de rota, (c) Comparação das rotas calculadas , (d) Rota final gerada .

Apesar das limitações dos dispositivos utilizados no desenvolvimento do trabalho, o que inviabiliza aplicações com restrições rígidas, este serviço apresentou um tempo de resposta aceitável, de aproximadamente 20 segundos à cada ciclo do algoritmo. Cada um desses ciclos equivale a 16 pares de imagens processadas, considerando a imagem base obtida previamente, referente à imagem inicial. Ou seja, o tempo necessário para processar cada par de imagens foi de aproximadamente 1 segundo e o atraso de inicialização da câmera igual a 4 segundos, resultando no tempo de resposta citado. Este resultado impôs algumas limitações ao sistema, influenciando na velocidade máxima permitida ao robô durante sua navegação, por exemplo, que foi de A

velocidade máxima utilizada foi de 150mm/s, enquanto a velocidade média foi de 125mm/s, aproximadamente. Embora haja diversos sistemas propostos na literatura que realizam planejamento de rota e desvio de obstáculo utilizando visão computacional, não é viável compará-los com o sistema proposto neste trabalho, pois o mesmo faz uso de um processador consideravelmente inferior para fazer o tratamento de imagens. A maioria dos sistemas foca em métodos eficientes para transferência da imagem capturada pela RSSF até uma estação base com alto poder de processamento onde a imagem será processada (Olsen & Hoover 1999). Deve-se enfatizar que nesta dissertação foi explorado o processamento de imagens na própria rede de sensores sem fio (processamento *in networking*).

A segunda etapa dos testes do serviço de auxílio à navegação foi realizada em ambiente real. Os experimentos foram realizados em um ambiente estruturado, sem interferências e de piso com baixo índice de atrito. As medidas referentes ao mapa do local foram obtidas a partir da planta baixa do ambiente. Utilizou-se um robô Pioneer 3-DX (P3DX) e um computador portátil como computador de bordo. O mapa do local utilizado nos testes é mostrado através da Figura 4.3.

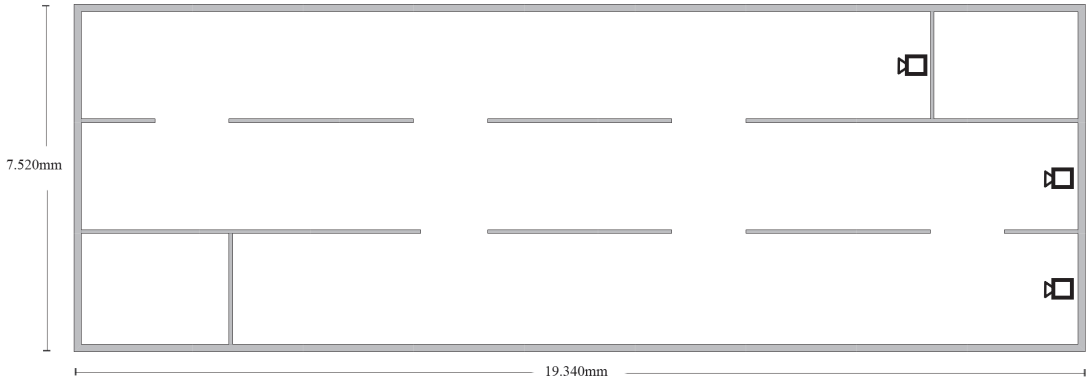


Figura 4.3: Mapa do ambiente de teste utilizado.

No teste realizado foi definido um ponto inicial e final fixos, de forma que a trajetória inicial calculada pelo robô fosse sempre a mesma. A rota calculada pelo robô é exibida através do tracejado na cor verde, Figura 4.4.

A partir da rota inicial, várias situações foram exploradas através da manipulação dos índices de congestionamento nos corredores a fim de forçar o robô a tomar diferentes decisões para assim

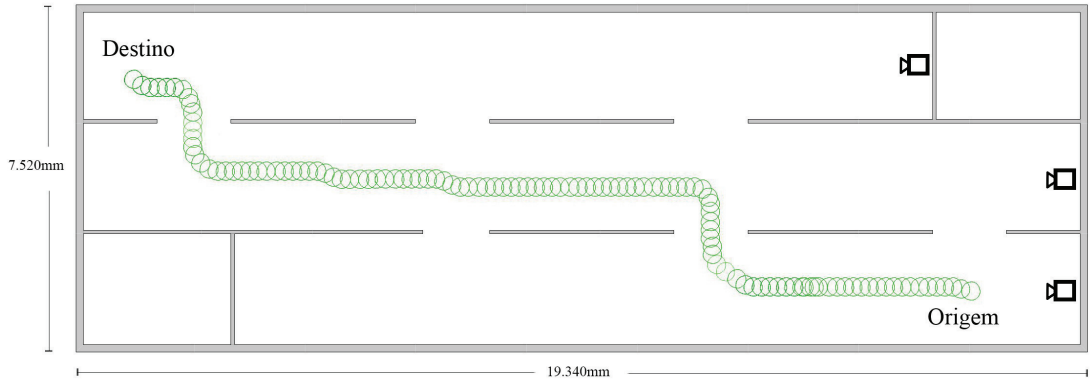


Figura 4.4: Rota inicial calculada através do algoritmo D*.

analisar seu comportamento ao longo da trajetória. Foram utilizados quatro nós sensores, três compostos por câmeras e posicionados estrategicamente pelo ambiente, e um, anexo ao robô e capaz de receber o resultado gerado pela RSSF. A Figura 4.5 indica que houve movimentação no corredor intermediário, fazendo com que o robô evitasse seguir por ele e optasse pelo corredor superior. O tracejado em vermelho corresponde ao caminho já percorrido pelo robô.

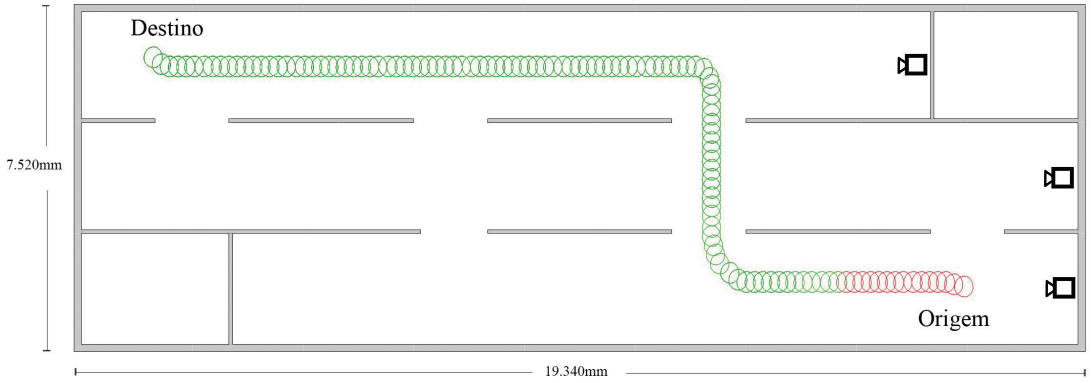


Figura 4.5: Recalculo de rota efetuado devido ao alto índice de movimentos no corredor intermediário.

No momento em que o robô optou seguir pelo caminho apresentado na Figura 4.5, pessoas passaram pelo corredor superior, aumentando seu peso e fazendo com o que o robô decidisse retornar e seguir pelo caminho apresentado na Figura 4.6, de acordo com o recalculo baseado no custo realizado pelo algoritmo D*.

Diante dos testes realizados pode-se dizer que o sistema é viável para ambientes com corredo-

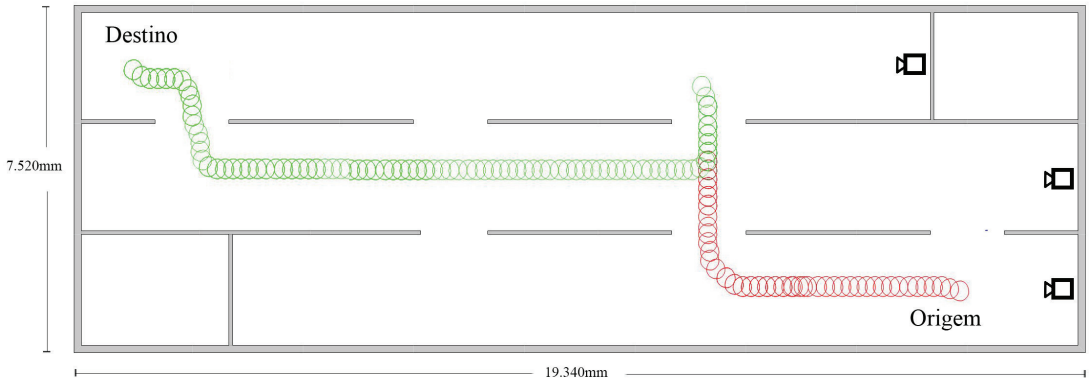


Figura 4.6: Recalculo de rota efetuado devido ao alto índice de movimentos no corredor superior.

res longos a partir de aproximadamente 8m, que é bastante comum em hospitais, por exemplo. Além disso, a velocidade utilizada é compatível com robôs assistivos transportando pessoas com deficiências de locomoção e dificuldades de atuação sobre o robô, como cadeiras de rodas robóticas, por exemplo.

4.2 Localização

O serviço de localização também foi testado em ambiente real. No experimento foi utilizado um robô Seekur Jr. (possui dimensões semelhantes as de uma cadeira de rodas robótica) que navegou entre os pontos definidos na Figura 4.7 enquanto nós equipados com sensores multimídia, espalhados pelo ambiente, ficaram responsáveis pela estimação da sua posição em cada um dos pontos. Inicialmente, foi utilizado um nó sensor A, capaz de abranger a marca anexa ao robô em todos os pontos da Figura 4.7. Em seguida, utilizaram-se outros dois nós sensores, B e C, para cobrir os pontos onde as estimações feitas pelo nó sensor A apresentaram as maiores diferenças em relação a posição real do robô nos respectivos pontos.

A marca utilizada no robô foi anexada a uma determinada altura de modo a evitar sua perda em casos de possíveis obstruções causadas por pessoas presentes no ambiente assim como para minimizar o erro de detecção da marca causado pela existência de outros objetos da mesma cor no campo de visão da câmera. Os nós da rede foram fixados à uma altura de 1.7m nas paredes, valor que corresponde à soma entre a altura do robô e a altura da haste utilizada para fixar a

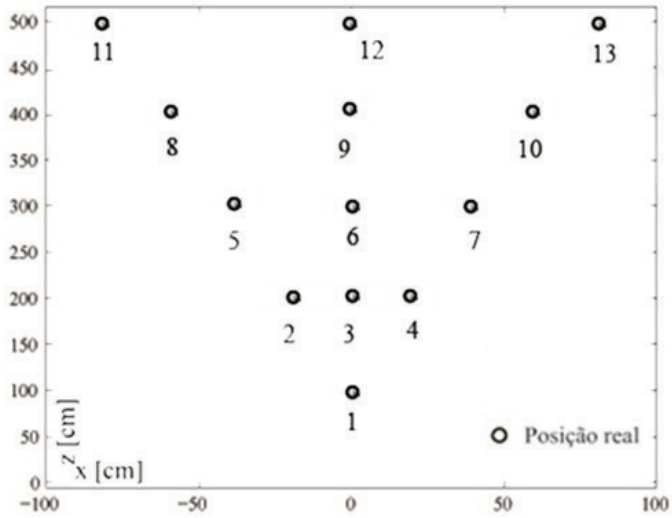


Figura 4.7: Pontos definidos com base nas medidas obtidas a partir da planta baixa do ambiente.

marca, o que permite o alinhamento da marca ao campo de visão das câmeras.

A análise do funcionamento do algoritmo foi feita com a estimação de posição sobre os pontos pré-definidos, baseados nas medidas fornecidas pela planta baixa do local. O robô foi guiado através de teleoperação para cada ponto, e em cada um deles uma requisição de localização foi executada. Dessa forma, a cada instante os nós realizaram a captura da imagem e, através da detecção da marca, estimaram a posição do robô naquele momento (posições nos eixos x e z, pois y é fixo). A Figura 4.8 apresenta os resultados para as estimativas nos 13 pontos sob a perspectiva do nó A, sendo que os pontos mais distantes (11, 12 e 13) estão aproximadamente a 5m de distância. O algoritmo de localização foi implementado com 483 linhas de código e ocupa 13kB de memória do nó sensor.

O erro médio total obtido a partir da perspectiva do nó A foi de aproximadamente 16cm, um valor relativamente pequeno dado as dimensões usuais de robôs móveis assistivos e o fato de que erros inferiores a 50cm são considerados aceitáveis para ambientes *indoor* (Long Cheng & Zhang 2011). Apesar do baixo erro médio observado, notou-se um elevado desvio padrão, sendo que, em uma distância de até 4m (pontos 1 a 7 na Figura 4.8) o erro médio foi de 7,3cm, enquanto nos pontos mais distantes, ou seja, a partir de 4m (pontos 8 a 13), este erro passou

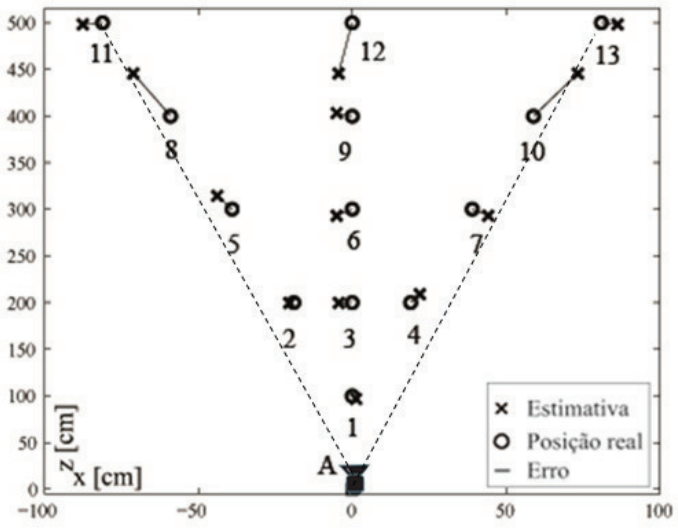


Figura 4.8: Estimativas da localização do robô sob a perspectiva do nó A.

a ser de 27,96cm. A Tabela 4.1 apresenta os resultados das estimativas. Se compararmos os resultados obtidos com os de sistemas que fazem uso de técnicas clássicas para estimação da localização, como a força do sinal recebido, por exemplo, os erros médios são inferiores, devido principalmente ao fato desses sistemas fazerem uso do canal de comunicação sem fio (troca de ondas de rádio) para estimação da posição do alvo. Por estarem sujeitos a fenômenos como perda de caminho, interferência, reflexão, sombreamento, *fading*, entre outros, o erro médio de estimação da posição do alvo é geralmente superior a 50cm e a melhoria desta estimação não pode ser garantida pelo aumento da quantidade de nós sensores (Paolo Pivato & Petri 2011).

Os erros estimados para cada eixo também foram analisados. O erro médio no eixo x foi de 5,59cm e no eixo z de 14,89cm. Separando os erros pelas regiões supracitadas, na região que abrange a partir do nó sensor 1 até o 7, os erros foram 3,59cm e 5,89cm, nos eixos x e z respectivamente, enquanto na região que abrange a partir do nó sensor 8 até 13, estes erros passaram para 7,93cm e 25,40cm. Estes resultados são apresentados na Tabela 4.2.

O maior erro nas estimativas após 4m se deve, além das limitações da câmera, ao efeito da variação da luminosidade sobre a profundidade da marca, dificultando a identificação da cor desejada. No cenário de ambientes assistivos é natural o uso de múltiplas câmeras, nesse caso as leituras feitas por nós a partir de outra perspectiva podem ser mais precisas pelo fato de

Tabela 4.1: Resultados do erro Euclidiano.

Parâmetro	Valor
Erro mínimo (1-13)	1,53cm
Erro máximo (1-13)	54,59cm
Erro médio (1-13)	16,84cm
Desvio padrão do erro (1-13)	19,18cm
Erro mínimo (1-7)	1,53cm
Erro máximo (1-7)	15,22cm
Erro médio (1-7)	7,30cm
Desvio padrão do erro (1-7)	4,63cm
Erro mínimo (8-13)	5,54cm
Erro máximo (8-13)	54,59cm
Erro médio (8-13)	27,96cm
Desvio padrão do erro (8-13)	24,11cm

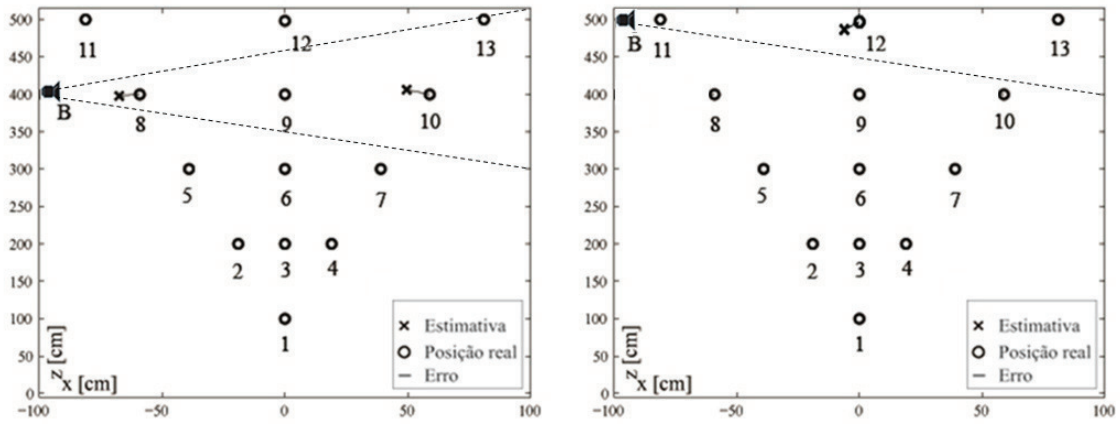
Tabela 4.2: Resultados do erro nos eixos.

Parâmetro	Valor
Erro médio no eixo x (1-13)	5,59cm
Erro médio no eixo z (1-13)	14,89cm
Desvio padrão no eixo x (1-13)	3,72cm
Desvio padrão no eixo z (1-13)	19,67cm
Erro médio no eixo x (1-7)	3,59cm
Erro médio no eixo z (1-7)	5,89cm
Desvio padrão no eixo x (1-7)	1,68cm
Desvio padrão no eixo z (1-7)	5,03cm
Erro médio no eixo x (8-13)	7,93cm
Erro médio no eixo z (8-13)	25,40cm
Desvio padrão no eixo x (8-13)	4,21cm
Desvio padrão no eixo z (8-13)	25,54cm

não estarem suscetíveis ao mesmo efeito. A Figura 4.9 apresenta as estimativas a partir da perspectiva de um nó B sobre os pontos que apresentaram maior erro sob a perspectiva do nó A. Sob essa perspectiva, o erro médio diminuiu, passando a ser de aproximadamente 7cm. A Tabela 4.3 apresenta os resultados das estimativas a partir desta perspectiva.

O erro médio das estimativas no eixo x não foi influenciado de forma significativa devido ao pequeno ângulo de abertura da câmera, porém, no eixo z houve uma diminuição considerável deste erro, que passou, em relação a perspectiva do nó A, de 14,89cm para 5,05cm. Estes resultados são apresentados na Tabela 4.4.

Com o uso de vários nós de forma redundante seria possível explorar técnicas de fusão ou



(a) (b)

Figura 4.9: Estimativas da localização do robô sob a perspectiva do nó B: (a) Estimativa da posição dos nós 8 e 10, (b) Estimativa da posição do nó 12.

Tabela 4.3: Resultados do erro Euclidiano.

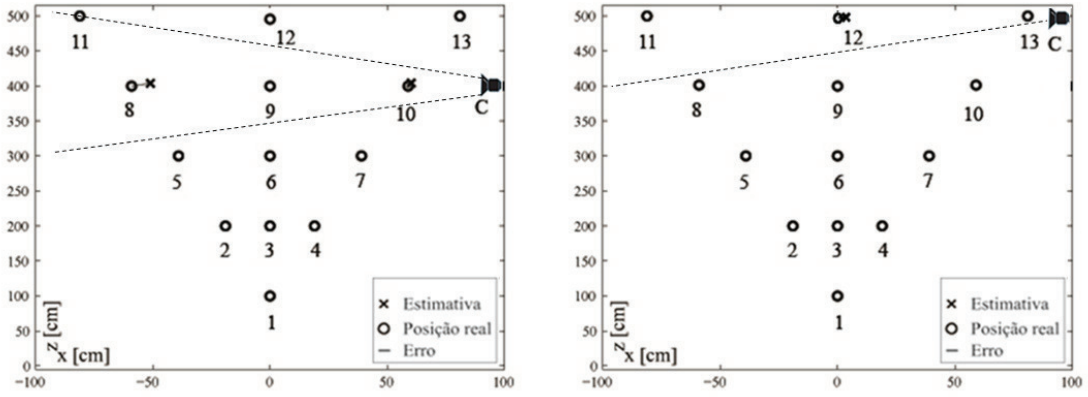
Parâmetro	Valor
Erro mínimo (1-13)	1,53cm
Erro máximo (1-13)	15,22cm
Erro médio (1-13)	7,75cm
Desvio padrão do erro (1-13)	3,70cm
Erro mínimo (1-7)	1,53cm
Erro máximo (1-7)	15,22cm
Erro médio (1-7)	7,30cm
Desvio padrão do erro (1-7)	4,63cm
Erro mínimo (8-13)	5,54cm
Erro máximo (8-13)	11,25cm
Erro médio (8-13)	8,28cm
Desvio padrão do erro (8-13)	2,57cm

até mesmo definir o grau de confiabilidade da estimação considerando a distância entre a marca e a câmera, onde aquelas mais próximas da marca teriam maior probabilidade de efetuar a estimação de forma precisa.

Através do cenário estabelecido foi ainda possível analisar as estimativas referentes à posição do robô a partir de uma terceira perspectiva, fornecida pelo nó C. A Figura 4.10 apresenta as estimativas a partir da perspectiva do nó C. Considerando essa perspectiva, os valores das estimativas obtidas são apresentadas na Tabela 4.5, onde o erro médio neste caso foi de aproximadamente 6cm.

Tabela 4.4: Resultados do erro nos eixos.

Parâmetro	Valor
Erro médio no eixo x (1-13)	5,09cm
Erro médio no eixo z (1-13)	5,05cm
Desvio padrão no eixo x (1-13)	2,38cm
Desvio padrão no eixo z (1-13)	4,17cm
Erro médio no eixo x (1-7)	3,59cm
Erro médio no eixo z (1-7)	5,89cm
Desvio padrão no eixo x (1-7)	1,68cm
Desvio padrão no eixo z (1-7)	5,03cm
Erro médio no eixo x (8-13)	6,85cm
Erro médio no eixo z (8-13)	4,08cm
Desvio padrão no eixo x (8-13)	1,84cm
Desvio padrão no eixo z (8-13)	3,06cm



(a) (b)

Figura 4.10: Estimativas da localização do robô sob a perspectiva do nó C: (a) Estimativa da posição dos nós 8 e 10, (b) Estimativa da posição do nó 12.

O erro médio das estimativas no eixo x, assim como o erro gerado pela estimativa sob a perspectiva do nó B, também não foi influenciado de forma significativa devido ao mesmo motivo citado anteriormente. Já no eixo z houve uma diminuição do erro para 4,42cm. Os resultados são apresentados na Tabela 4.6.

Esse capítulo apresentou os experimentos realizados para validar os serviços propostos neste trabalho. Através dos resultados obtidos a partir desses experimentos foi possível analisar a viabilidade do sistema através do seu desempenho e efetividade no cumprimento de seu objetivo. O capítulo seguinte irá concluir este trabalho além de abordar possíveis melhorias e

desenvolvimentos para sua continuidade.

Tabela 4.5: Resultados do erro Euclidiano.

Parâmetro	Valor
Erro mínimo (1-13)	1,53cm
Erro máximo (1-13)	15,22cm
Erro médio (1-13)	6,71cm
Desvio padrão do erro (1-13)	3,51cm
Erro mínimo (1-7)	1,53cm
Erro máximo (1-7)	15,22cm
Erro médio (1-7)	7,30cm
Desvio padrão do erro (1-7)	4,63cm
Erro mínimo (8-13)	4,31cm
Erro máximo (8-13)	8,91cm
Erro médio (8-13)	6,01cm
Desvio padrão do erro (8-13)	1,67cm

Tabela 4.6: Resultados do erro nos eixos.

Parâmetro	Valor
Erro médio no eixo x (1-13)	4,26cm
Erro médio no eixo z (1-13)	4,42cm
Desvio padrão no eixo x (1-13)	2,03cm
Desvio padrão no eixo z (1-13)	4,01cm
Erro médio no eixo x (1-7)	3,59cm
Erro médio no eixo z (1-7)	5,89cm
Desvio padrão no eixo x (1-7)	1,68cm
Desvio padrão no eixo z (1-7)	5,03cm
Erro médio no eixo x (8-13)	5,03cm
Erro médio no eixo z (8-13)	2,71cm
Desvio padrão no eixo x (8-13)	2,28cm
Desvio padrão no eixo z (8-13)	1,32cm

Conclusões e Perspectivas

O capítulo anterior apresentou os resultados gerados a partir da execução dos serviços desenvolvidos neste trabalho. Este capítulo apresenta as conclusões obtidas a partir desses resultados, destacando as contribuições e possíveis melhorias para trabalhos futuros.

Contribuições

O trabalho desenvolvido propôs o uso de plataformas *high-end* em sistemas de RSSF para implementação de aplicações mais complexas e sofisticadas do que aquelas cujos sistemas fazem uso de plataformas *low-end*. As aplicações implementadas neste trabalho foram apresentadas como serviços oferecidos pelo ambiente, que foi equipado com plataformas *high-end*, equipadas com sensores multimídia. Com o uso de nós sensores de alta capacidade foi possível realizar operações de tratamento de imagens (obtidas a partir dos sensores multimídia), o que possibilitou a geração de estimativas referentes à localização de uma marca anexa ao robô e à taxa de ocupação dos corredores de um ambiente, técnica conhecida como *crowd detection*. Esses serviços se encarregaram de oferecer assistividade à um robô móvel no cumprimento de seu objetivo. Toda a comunicação da RSSF foi efetuada através de uma versão simplificada do protocolo AODV, que foi desenvolvida para possibilitar o roteamento de mensagens entre nós afastados geograficamente. Através desse cenário foi possível transferir o controle e a inteligência do sistema para o ambiente ao invés de centralizá-los em uma única entidade, como um robô móvel por exemplo. Esse paradigma permitiu a utilização de toda uma variedade de recursos que um

ambiente pode suportar, possibilitando a obtenção de informações sobre o estado de um local antes do mesmo ter sido explorado, o que, consequentemente, torna o sistema mais eficiente em relação aos sistemas tradicionais.

Perspectivas

Uma melhoria significativa ao trabalho desenvolvido seria a otimização do código dos nós sensores através do uso de instruções oferecidas pelo coprocessador do Imote2, criadas especificamente para possibilitar o aumento do desempenho de processamento de dados multimídia. Através disso o tempo de resposta do sistema reduziria consideravelmente, tornando o sistema proposto ainda mais viável para implentações reais. Neste novo cenário, vários serviços adicionais poderiam ser propostos e interfaces para acessá-los poderiam ser desenvolvidas. Embora os serviços desenvolvidos tenham apresentado resultados aceitáveis sob ambiente controlado, algumas limitações relacionadas ao desempenho foram recorrentes, devido principalmente às restrições de processamento e taxa de transmissão ainda comuns nas plataformas de RSSF. Apesar dessas plataformas já terem tido grande avanço nos últimos anos, ainda estão bastante distantes daquelas utilizadas em redes *ad hoc*. Espera-se que nos próximos anos a tecnologia de semicondutores avance o suficiente para permitir a queda do custo, aumento de capacidades e minimização do consumo de energia.

Publicações

TEIXEIRA, Fábio V., SOUZA, Ricardo S., OLIVI, Leonardo R., ROHMER, Eric, GUIMARÃES, Eliane, CARDOZO, Eleri. Infraestrutura de Rede de Sensores Sem Fio para Ambientes Assistivos. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 19, 2012, Campina Grande. Anais da Sociedade Brasileira de Automação. Campina Grande: SBA, 2012.

SOUZA, Ricardo S., OLIVI, Leonardo R., ROCHA, Lúcio A., RODRIGUES, Diego, TEIXEIRA, Fábio V., GUIMARÃES, Eliane, CARDOZO, Eleri. Controle de Robôs Móveis Através de Redes de Sensores Sem Fio. In: CONGRESSO BRASILEIRO DE REDES DE COMPU-

TADORES E SISTEMAS DISTRIBUÍDOS, 29, 2011, Campo Grande. Anais da Sociedade Brasileira de Redes de Computadores. Campo Grande: SBC, 2011.

Bibliografia

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. & Cayirci, E. (2002). Wireless sensor networks: a survey, *Computer Networks* **38**: 393–422.

Akyildiz, I. F. & Vuran, M. C. (2010). *Wireless Sensor Networks*, Wiley.

Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgeson, A. & Han, R. (2005). Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms, *International Journal of Computer Science & Engineering Survey* **10**: 563–579.

Bonifácio, T. G. (2010). *Implementação de um protocolo mesh multi-hop baseado em algoritmo de roteamento geográfico para redes de sensores sem fio*, Master's thesis, Escola de Engenharia de São Carlos, Universidade de São Paulo.

Boulis, A. (2011). Castalia: A simulator for wireless sensor networks and body area networks, `http://castalia.npc.nicta.com.au/pdfs/Castalia%20-%20User%20Manual.pdf`. Acessado em: 05/05/12.

Cao, Q., Abdelxaher, T., Stankovic, J. & He, T. (2008). The liteos operating system: Towards unix-like abstrations for wireless sensor networks, *International Conference on IEEE*, pp. 233–244.

Crossbow (2007). Imote2 hardware reference manual, http://web.univ-pau.fr/~cpham/ENSEIGNEMENT/PAU-UPPA/RESA-M2/DOC/Imote2_Hardware_Reference_Manual.pdf. Acessado em: 15/03/12.

Dargie, W. & Poellabauer, C. (2010). *Fundamentals of Wireless Sensor Networks Theory and Practice*, Wiley.

Farooq, M. O. & Kunz, T. (2011). Operating systems for wireless sensor networks: A survey, *Sensors* **11**: 5900–5930.

Forster, A. & Murphy, A. L. (2010). A critical survey and guide to evaluating wsn routing protocol. http://www.dti.supsi.ch/~afoerste/publications/foerster-conet2010.pdf. Acessado em: 22/01/13.

FTDI (2012). Future technology devices international ltd., http://www.ftdichip.com/. Acessado em: 20/12/12.

Gregory, W. (2012). Open computer vision library - opencv, http://opencvlibrary.sourceforge.net/. Acessado em: 13/11/12.

Guanling Chen, Prabhu Govindaswamy, N. L. & Wang, J. (2008). Continuous camera-based monitoring for assistive environments, *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments*, New York, NY, USA, pp. 1–8.

Gutierrez, J. A., Naeve, M., Callaway, E., Bourgeois, M., Mitter, V. & Heile, B. (2001). Ieee 802.15.4: A developing standard for low-power low-cost wireless personal area networks, *Network* **15**: 12–19.

He, Y., Shen, X., Liu, Y., Mo, L. & Dai, G. (2010). Listen: Non-interactive localization in wireless sensor camera networks, *Proceedings of the IEEE Real-Time Systems Symposium*, San Diego, CA, USA, pp. 205–214.

Hill, J., Horton, M., Kling, R. & Krishnamurthy, L. (2004). The plataform enabling wireless sensor networks, *Communications of the ACM* **47**: 41–46.

Karl, H. & Willig, A. (2005). *Protocols and Architectures for Wireless Sensor Networks*, Wiley.

Kasteleiner, J. (2010). Principles of applying embedded linux on imote2. `http://jkastel.homepage.t-online.de/thesis.pdf`. Acessado em: 12/18/12.

Kellner, A., Behrends, K. & Hogrefe, D. (2010). Simulation environments for wireless sensor networks, *Technical report*, Georg-August-Universitat Gottingen.

Kinney, P. (2003). Zigbee technology: Wireless control that simply works, *Communications Design Conference*, pp. 1–20.

Kosmopoulos, D. I. (2010). Behavior monitoring for assistive environments using multiple views, `http://users.iit.demokritos.gr/~dkosmo/downloads/comvis/uais10/UAIS10.pdf`. Acessado em: 27/01/14.

Levis, P. & Le, N. (2003). Tossim: A simulator for tinyos networks, *Technical report*, Berkeley University.

Lin, Q., Zeng, X., Jiang, X. & Jin, X. (2011). Video notes based collaborative object localization framework in wireless multimedia sensor networks, *Advanced Materials Research: Equipment Manufacturing Technology and Automation* pp. 1078–1083.

Liu, L., Zhang, X. & Ma, H. (2010). Optimal node selection for target localization in wireless camera sensor networks, *IEEE Transactions on vehicular technology* **59**(7).

Lohier, S., Rachedi, A., Livolant, E. & Salhi, I. (2011). Wireless sensor network simulators relevance compared to a real ieee 802.15.4 testbed, *7th International Wireless Communications and Mobile Computing Conference*, pp. 1347–1352.

Long Cheng, C.-D. W. & Zhang, Y.-Z. (2011). Indoor robot localization based on wireless sensor networks, *IEEE Transactions on Consumer Electronics* **57**.

Margi, C., Petkov, V., Obraczka, K. & Manduchi, R. (2006). Characterizing energy consumption in a visual sensor network testbed, *Testbeds and Research Infrastructures for the Development of Networks and Communities*, pp. 8 pp.–339.

Meger, D., Marinakis, D., Rekleitis, I. & Dudek, G. (2009). Inferring a probability distribution function for the pose of a sensor network using a mobile robot, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 756–762.

Mehta, V., Sheng, W., Chen, T. & Shi, Q. (2009). Development and calibration of a low cost wireless camera sensor network, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, pp. 110–115.

MobileSim (2011). Mobilesim - mobilerobot's research and academic custumer support, `http://robots.mobilerobots.com/wiki/MobileSim`. Acessado em: 15/01/12.

Olivi, L., Souza, R., Paolieri, F., Guimaraes, E. & Cardozo, E. (2012). A distributed navigation strategy for mobile robots based on wireless sensor networks, *Brazilian Conference on Automation*, pp. 1–8.

Olsen, B. D. & Hoover, A. (1999). Path planning for mobile robots using a video camera network, *Advanced Intelligent Mechatronics*, pp. 890–895.

OpenOCD (2012). Open on-chip debugger, `http://openocd.sourceforge.net/`. Acessado em: 19/03/12.

Paolo Pivato, L. P. & Petri, D. (2011). Accuracy of rss-based centroid localization algorithms in an indoor environment, *IEEE Transaction on Instrumentation and Measurement* **60**.

Perkings, C. (2003). Ad hoc on-demand distance vector (aodv) routing, `http://www.ietf.org/rfc/rfc3561.txt`. Acessado em: 15/07/12.

Pottie, G. J. (2001). Wireless integrated network sensors (wins): The web gets physical. `http://www.seas.ucla.edu/~pottie/papers/nae_01.pdf`. Acessado em: 15/03/13.

Rabaey, J. M., Ammer, M. J., Patel, D. & Roundy, S. (2000). Picoradio supports ad hoc ultra-low power wireless networking, *Computer* pp. 42–48.

Rekleitis, I., Meger, D. & Dudek, G. (2006). Simultaneous planning, localization, and mapping in a camera sensor network, *Robotics and Autonomous Systems Journal* **54**: 921–932.

Shio Kumar Singh, M. P. S. & Singh, D. K. (2010). Routing protocols in wireless sensor networks - a survey, *International Journal of Computer Science & Engineering Survey* **1**: 63–83.

Souza, R. (2011). *Uma plataforma para integração de redes de sensores sem fio a aplicações de robótica móvel*, Master's thesis, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas.

Srivastava, M., Muntz, R. & Potkonjak, M. (2001). Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments, *Proceedings of the International Conference on Mobile Computing and Networking*, Rome, ITA, pp. 132–138.

Stentz, A. (1995). The focussed d* algorithm for real-time replanning, *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1652–1659.

Svenstrup, M., Bak, T. & Andersen, H. J. (2010). Trajectory planning for robots in dynamic human environments, *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4293–4298.

Thorup, R. E. (2007). *Implementing and evaluating the dymo routing protocol*, Master's thesis, Department of Computer Science, University of Aarhus.

Vangelis Metsis, Zhengyi Le, Y. L. & Makedon, F. (2008). Towards an evaluation framework for assistive environments, *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments*, pp. 1–8.

Watteyne, T. (2008). *Energy-Efficient Self-Organization for Wireless Sensor Network*, PhD thesis, Institut National des Sciences Appliquées de Lyon.