

Luiz Rogério Batista De Pieri

**Monitoração de Pressão Sonora Utilizando Redes de Sensores
Sem Fio – SPLDM**

Florianópolis – SC, Brasil

Julho de 2008

Luiz Rogério Batista De Pieri

Monitoração de Pressão Sonora Utilizando Redes de Sensores Sem Fio - SPLDM

Trabalho de conclusão de curso apresentado como
parte dos requisitos para obtenção do grau de
Bacharel em Sistemas de Informação

Orientador:

Prof. Dr. Mário Antônio Ribeiro Dantas

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Florianópolis – SC, Brasil

Julho de 2008

Monografia de Projeto Final de Graduação sob o título “*Monitoração de Pressão Sonora Utilizando Redes de Sensores Sem Fio – SPLDM*”, defendida por Luiz Rogério Batista De Pieri e aprovada em Julho de 2008, em Florianópolis, Estado de Santa Catarina, pela banca examinadora constituída pelos professores:

Prof Ph.D. Mário Antônio Ribeiro Dantas
Orientador

Prof. Dr. Antônio Augusto Medeiros Fröhlich
Universidade Federal de Santa Catarina

Prof. MSc. Alex Sandro Roschildt Pinto
Universidade Federal de Santa Catarina

*Dedico este trabalho
à minha esposa Fernanda,
a meus pais Luiz e Neuza,
minhas irmãs Luciana e Jakeline,
minha avó Camila,
aos meus tios Daniel, Regina,
Mosa, Joel, Jurema, Mari e Aguiar
e ao Dr. Wagner Machado
que sempre acreditaram em mim,
motivando-me a chegar até aqui.*

Agradecimentos

Agradeço primeiramente a Deus que é fonte de toda minha inspiração. Meus sinceros agradecimentos se estendem:

- a meus pais Luiz e Neuza que sempre me motivaram a buscar o conhecimento;
- a minhas irmãs Luciana e Jakeline por seus sorrisos motivadores e a meu cunhado José Carlos por cuidar bem de minha irmã e por eu poder chamá-lo de amigo;
- ao professor doutor Mário Dantas por seu incentivo e orientação;
- ao Laboratório de Pesquisa em Sistemas Distribuídos – LaPeSD, pelo fornecimento de equipamentos e espaço para o desenvolvimento deste trabalho e ao Hendri por sempre garantir que os equipamentos estivessem disponíveis;
- a meus amigos Pr. Rudi e sua esposa Pra. Kátia por sua dedicação e atenção sempre que precisei;
- a equipe de áudio da Igreja Batista Palavra Viva e ao ministério Avivah;
- aos meus amigos André Luiz de Souza e Rafael Carlos Ferreira, motivadores natos sempre dispostos a colaborar, e Allan Celestino por me apresentar ao curso de Sistemas de Informação;
- os grupo dos NERDS SIN com ênfase para Cristina Orthmann, Luciano Antonio Costa e Marcos H. Santos pela contínua troca de conhecimento e crescente amizade;
- aos membros do Grupo 7, por abrirem um espaço em suas agendas para os almoços de sábado;
- ao Dr. Wagner Machado por acreditar em meu potencial;
- em especial a minha esposa Fernanda por sua motivação, amor, carinho, dedicação e por compreender minha ausência e meus momentos de tensão, a ela não apenas meu agradecimento, mas também meu coração.

Resumo

Avanços nas comunicações sem fio e componentes eletrônicos em geral, tem levado ao desenvolvimento de redes de baixo custo e baixo consumo de energia, exemplo de tal configuração são as redes de sensores multifuncionais, que vêm recebendo uma crescente atenção. Redes de sensores sem fio (RSSF) vêm sendo utilizadas nos mais variados ambientes e com os mais variados fins. Entre os campos de utilização das RSSF estão as engenharias em geral, na agricultura e na monitoração ambiental, em aplicações militares e clínicas [HAENGGI, 2005].

Este trabalho apresenta uma proposta de utilização de sensores sem fio afim de apoiar o trabalho com relação ao áudio por meio da monitoração da pressão sonora (SPL – *Sound Pressure Level*) no ambiente da sonorização. O objetivo principal está em apresentar ao técnico de áudio uma leitura da distribuição da pressão sonora no ambiente sem que o mesmo precise deslocar-se utilizando um decibelímetro, instrumento utilizado para medição de SPL, como de costume.

Ao final dos experimentos verificou-se que é possível monitorar com sucesso pressões sonoras dentro de determinados limites utilizando RSSF. Para alcançar a utilização em campo por técnicos de áudio no entanto, faz-se necessário ampliação do escopo do trabalho e principalmente hardware com recursos mais elaborados.

Palavras-chave: RSSF, Redes, Sensores, Sem Fio, SPL, Decibelímetro, medição sonora, SPL

Abstract

Technical progress on wireless study field and electronics components improvements has aimed the development of low cost and low energy consumption networks. As instance of this trend are the WSN (Wireless Sensor Networks) that have been utilizing in the most heterogenic environments for all sort of purposes. The WSN use extends over the fields of engineering, agriculture, environmental monitoring, military applications and medical clinics [HAENGGI, 2005].

This paper presents a sensor utilization proposal with purpose to support audio technician in sound pressure level measurement daily basis tasks. The main purpose is to display to audio technician the sound pressure measurements without the necessity of the physically dislocation around the sound test environment with a decibel meter in hands.

After all tests, the results gotten showed the feasibility of sound pressure measure among specific limits using Wireless Sensor Networks. However, in order to have a professional use, it is necessary to review the scope of this project altogether, and moreover, to acquire hardware components with more refined features.

Keywords: WSN, Network, Sensor, Wireless, SPL, decibelimeter, sound mesure, SPL

Sumário

1. Introdução.....	1
1.1 Motivação.....	2
1.1 Objetivo Geral	2
1.3 Objetivos Específicos.....	2
1.4 Justificativa.....	3
1.5 Delimitação do Escopo.....	3
1.6 Organização do Documento.....	3
2 Redes Sem Fio.....	4
2.1 Redes de Sensores Sem Fio (RSSF).....	4
2.1.1 Componentes de RSSF.....	5
2.1.1.1 Nodos Sensores e Interfaces de Comunicação Sem Fio.....	5
2.1.1.2 Nodos Gateway.....	6
2.1.2 Características das Redes de Sensores Sem Fio.....	7
2.1.3 Protocolos de Comunicação em Redes de Sensores Sem Fio.....	8
2.1.3.1 O Protocolo S-MAC.....	9
2.1.3.2 Padrão IEEE 802.15.4 e ZigBee.....	10
2.1.4 TinyOS e nesC.....	10
2.1.4.1 nesC.....	10
3 Ondas Sonoras, Sistemas de Sonorização e Sensores para Medição de Som.....	12
3.1 Ondas Sonoras.....	12
3.1.1 Efeitos da temperatura sobre as ondas sonoras.....	12
3.1.2 Efeitos da Umidade e outros fatores.....	12
3.2 Sistemas de Sonorização.....	13
3.2.1 Tipos de Microfones.....	15
3.2 Medição de Pressão Sonora (decibéis).....	16
3.3.1 Medições Aplicáveis a Sistemas de Sonorização.....	19
3.4 Efeitos do ruído sobre o organismo humano.....	21
3.5 Sensores para medição de som.....	22
4 SPLDM - Monitoração de Pressão Sonora utilizando Redes de Sensores Sem Fio.....	24
4.1 SPLDM – Sound Pressure Level Distributed Meter.....	25
4.2 Trabalhos Relacionados.....	29
5 Ambiente, Implementação e Resultados Experimentais.....	31
5.1 Ambiente de Desenvolvimento.....	31

5.1.1 Ferramentas.....	31
5.1.3 Hardware.....	31
5.2 Implementação.....	32
5.2.1 Nodos.....	32
5.2.1.1 Nodos Sensores.....	32
5.2.1.2 Nodo Base.....	33
5.2.2 Aplicação Java.....	33
5.3 Resultados Experimentais.....	33
5.3.1 Coleta de Dados para Calibração.....	34
5.3.2 Medições.....	37
6 Conclusões e Trabalhos Futuros.....	40
6.1 Conclusões.....	40
6.2 Trabalhos Futuros.....	40
Referências.....	42
Anexos.....	44
Código fonte da aplicação Java.....	44
Código Fonte da Aplicação nesC.....	50

Índice de ilustrações

Figura 1 - Transmissão multi-hop em uma RSSF [LOUREIRO et al., 2003].....	5
Figura 2 - Hardware básico de um nodo sensor [LOUREIRO et al., 2003].....	6
Figura 3 - Modelo genérico de uma RSSF com um gateway [LOUREIRO et al., 2003].....	7
Figura 4 - Modelo de RSSF com um nodo sink [LOUREIRO et al., 2003].....	7
Figura 5 - Mesas e caixas de som, amplificadores e microfones.....	13
Figura 6: Contornos de Mesma Audibilidade [CYSNE, 1997].....	17
Figura 7: Curvas de Compensação A, B e C [CYSNE, 1997].....	18
Figura 8 - Ultra Curve Behringer.....	20
Figura 9 - Decibelímetros.....	20
Figura 10: Plataforma e Placa de Sensores.....	23
Figura 11: SBT80 conectada a uma plataforma TelosB.....	23
Figura 12 - Ambiente para sonorização.....	25
Figura 13: Rede de sensores para monitorar um ambiente sonorizado.....	26
Figura 14: Diagrama Lógico - SPLDM.....	27
Figura 15: Diagrama de Seqüência.....	29
Figura 16: Registros dos sensores para pressões sonoras de 60, 70, 80, 90 e 100dB(A)	34
Figura 17: Comparativo entre pressão sonora e o registro dos sensores 1, 2 e 3.....	35
Figura 18: Geração de Linha de Tendência com o Microsoft Excel.....	36
Figura 19: Opções da Geração de Linha de Tendência.....	36
Figura 20: Linhas originais e de tendência dos sensores 1, 2 e 3.....	37
Figura 21: Telas do SPLDM com medições de 60dB(A), 70dB(A), 80dB(A), 90dB(A) e 100dB(A).....	38
Figura 22: Comparativo Medido x Esperado.....	39

Índice de Tabelas

Tabela 1: Relação entre atividade e nível de ruído.....	19
Tabela 2: Máxima exposição diária permissível.....	22
Tabela 3: Correlação dos Trabalhos Relacionados.....	30
Tabela 4: Média e desvio padrão das medições dos sensores.....	35
Tabela 5: Função e R^2 das Linhas de Tendência.....	37
Tabela 6: Interligação dos sensores à interface.....	38

1. Introdução

O papel de um técnico de áudio é fazer uso dos equipamentos disponíveis para levar o produto gerado por um CD/DVD, um palestrante ou por um grupo musical ao público ouvinte que deve sentir-se em um ambiente confortável. Para obter sucesso em seu trabalho o técnico precisa ter controle sobre o som propagado no ambiente de forma a deixá-lo o mais agradável possível para toda a gama de ouvintes do local.

Um dos aspectos essenciais a ser observado pelo técnico é o nível da pressão sonora, popularmente conhecido como volume do som. Em grandes locais não é uma tarefa simples monitorar a pressão sonora em toda a extensão deste ambiente. Entre os objetivos do projeto de um sistema de sonorização está a melhor distribuição da pressão sonora no ambiente, sendo assim, caixas de som são distribuídas sendo fixadas e direcionadas de forma estratégica buscando atingir este propósito [CYSNE, 1997].

Segundo [FERENCE, 1978], aspectos como temperatura e umidade podem modificar a velocidade de ondas sonoras, sendo assim, mudanças bruscas em curto espaço de tempo podem trazer dificuldades ao técnico de áudio para manter equilibradas as frequências do som no ambiente. A reverberação é outra característica das ondas sonoras que merece observação, ela está diretamente ligada à intensidade do som e aos materiais presentes no ambiente e seus respectivos coeficientes de absorção.

Este trabalho propõe uma rede de sensores sem fio (RSSF) que monitora o ambiente em diferentes pontos e apresenta ao técnico de áudio uma leitura da distribuição da pressão sonora no local. Com esta leitura é possível ao técnico corrigir esta distribuição de pressão no decorrer do evento sem a necessidade de deslocar-se no ambiente buscando reconhecer a condição do som nos diferentes pontos do local.

As RSSF vêm ganhando espaço nos mais diversos ambientes de influência tecnológica com as mais diversas funções. Na maioria das situações a função das RSSF é apoiar interações do tipo homem-ambiente, máquina-ambiente ou até mesmo máquina-homem. Este trabalho apresenta uma forma de apoiar a interação do homem com o ambiente e ampliar a eficiência de seu trabalho.

1.1 Motivação

Quando o indivíduo se dispõe a buscar conhecimento em áreas distintas, a tendência é que este encontre um ponto de convergência nos seus estudos. A experiência atuando com sonorização, a formação técnica em eletrônica e os novos conhecimentos adquiridos na academia de Sistemas de Informação motivaram o desenvolvimento de uma proposta que espera-se, possa somar para a comunidade de tecnologia como um todo.

1.1 Objetivo Geral

Monitorar a pressão sonora em um ambiente sonorizado para apoiar o trabalho do técnico de áudio

1.3 Objetivos Específicos

Utilizando uma rede de sensores, busca-se um modelo que possibilite a monitoração de um ambiente sonorizado no que diz respeito à distribuição da pressão sonora neste ambiente gerada por um sistema de sonorização. Para alcançar este objetivo serão utilizados sensores sem fio equipados com microfones, juntos estes sensores formam uma rede que envia dados a um computador que por sua vez apresenta ao usuário a leitura efetuada do ambiente.

Com o desenvolvimento deste projeto espera-se:

- Adquirir conhecimentos relacionados à programação de sensores sem fio, bem como aprender um novo paradigma de programação utilizando-se da linguagem nesC;
- Ampliar os conhecimentos de programação utilizando Java;
- Ampliar os conhecimentos sobre acústica e ondas sonoras;
- Ampliar os conhecimentos sobre integração de tecnologias;
- Ampliar os conhecimentos sobre transformação de grandezas físicas;

1.4 Justificativa

O avanço tecnológico está a cada dia mais automatizando e os procedimentos executados pelo homem. A operação de um sistema de som é um exemplo de área que merece atenção no que diz respeito à automação de processos. Atualmente é essencial a presença do técnico para a operação de um sistema de som em um evento, entretanto, a utilização de um sistema que permitisse a captura do som no ambiente do evento e sua transmissão até o local onde encontra-se o técnico, permitiria uma operação remota deste sistema. Entre as variáveis a serem observadas neste ambiente e transmitidas ao técnico, estaria a distribuição da pressão sonora no local, o foco de estudo deste trabalho.

1.5 Delimitação do Escopo

Este trabalho não aborda aspectos de segurança e conservação de energia considerados importantes no projeto de aplicações utilizando redes de sensores sem fio. No que diz respeito à análise sonora, alguns limites são encontrados em função de limitação de hardware principalmente ligados à simplicidade do microfone existente nos sensores em estudo.

1.6 Organização do Documento

Este documento está dividido basicamente em 6 capítulos, onde o primeiro capítulo apresenta a motivação e os objetivos a serem alcançados com o seu desenvolvimento, a seguir, no capítulo 2, são apresentadas informações pertinentes sobre redes sem fio com foco principal nas RSSF. O capítulo 3 apresenta ao leitor informações relacionadas a ondas sonoras e sistemas de reforço sonoro, citados como sistemas de sonorização. No quarto capítulo é apresentada a proposta do SPLDM (Sound Pressure Level Distributed Meter) e alguns trabalhos relacionados. O quinto capítulo demonstra os experimentos efetuados com o SPLDM no LaPeSD (Laboratório de Pesquisa em Sistemas Distribuídos) utilizando os equipamentos e espaço emprestados pela Universidade Federal de Santa Catarina (UFSC). O capítulo 6 contém as conclusões sobre os experimentos e os trabalhos futuros.

2 Redes Sem Fio

Redes sem fio representam importantes recursos nos ambientes corporativos. Seu diferencial está presente no baixo custo de infra-estrutura e no suporte a aplicações móveis. Em função do conforto e agilidade proporcionada aos usuários finais principalmente, as redes sem fio vêm sendo utilizadas por profissionais da mais diversas áreas tais como: corretores de bolsas de valores, inspetores de fábricas, representantes comerciais, entre outros [DANTAS, 2002].

Atualmente, já tornou-se muito comum a presença de redes sem fio fora do ambiente corporativo, estes recursos vêm sendo encontrados em praças de alimentação de shopping centers, condomínios ou ainda em residências, em sua maioria com o foco no acesso à rede mundial de computadores, a Internet.

2.1 Redes de Sensores Sem Fio (RSSF)

Cada vez mais a tecnologia está buscando trazer conforto ao usuário e diminuir a necessidade de infra-estrutura física para a implantação de sistemas. As redes sem fio vêm colaborando com estes aspectos no que diz respeito a dar mobilidade ao usuário e reduzir cada vez mais a necessidade de cabos. Telefones celulares, palm-tops e notebooks são os equipamentos mais populares utilizando-se desta tecnologia que continua em ascensão.

Segundo [LOUREIRO et al, 2003], os avanços nas áreas de micro-processadores, novos materiais de sensoriamento e a comunicação sem fio, vêm estimulando o desenvolvimento cada vez mais rápido de componentes miniaturizados com grande número de funções, é o caso dos sensores sem fio que compõem as redes de sensores sem fio (RSSF).

RSSF compõem uma fatia de sistemas embarcados que vem sendo utilizada em diversas áreas como militar, automação industrial e residencial, distribuição de água entre outros [ZHAO et al., 2003].

Vários aspectos diferenciam as RSSF das redes de computadores tradicionais entre eles estão [LOUREIRO et al., 2003]:

- Elevado número de nodos (nodo neste trabalho será considerado como um elemento computacional com capacidade de processamento, memória, interface de comunicação sem fio, além de um ou mais sensores de um mesmo tipo ou não);
- Nodos com restrições de energia com mecanismos para auto-configuração;
- RSSF são autônomas e requerem alto grau de interação para executar tarefas;

2.1.1 Componentes de RSSF

Os principais componentes das RSSF são nodos sensores, interfaces de comunicação sem fio e nodos para comunicação com outras entidades.

2.1.1.1 Nodos Sensores e Interfaces de Comunicação Sem Fio

Nodos sensores são dispositivos autônomos equipados com capacidade de sensoriamento, processamento e comunicação. Quando estes nodos são dispostos em rede em um modo *Ad Hoc*, formam as redes de sensores. Os nodos coletam dados via sensores, processam localmente ou coordenadamente entre vizinhos podendo enviar a informação para o usuário ou, em geral, para um *data sink*. Sendo assim, um nodo na rede tem tarefas diferentes: sensoriamento do ambiente, processamento da informação e tarefas associadas com o tráfego em um esquema de retransmissão *multi-hop*, como ilustra a figura 1 [LOUREIRO et al., 2003].

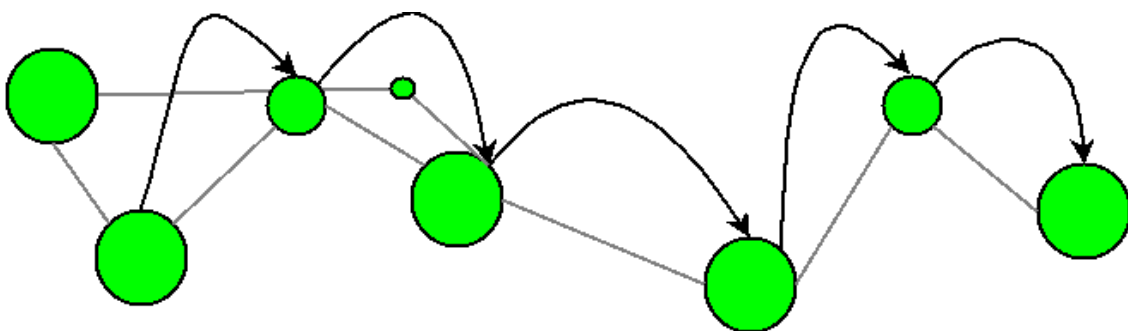


Figura 1 - Transmissão *multi-hop* em uma RSSF [LOUREIRO et al., 2003]

Um nodo micro-sensor é formado basicamente por: memória, transceptor, processador, sensor e bateria. Estes nodos são componentes de tamanho reduzido, são

reduzidas também suas capacidades no que diz respeito à fonte de energia (bateria), processamento e memória. A figura 2 traz um diagrama com as partes que compõem o nodo micro-sensor. O transceptor de um nodo micro-sensor é a sua interface de comunicação sem fio.

Em alguns casos, os nodos de uma RSSF desempenham a função de modificar valores do meio, afim de corrigir falhas e controlar o objeto monitorado. Nesse caso tem-se os atuadores. Quando um nodo sensor possui duas as funções, o dispositivo que implementa as mesmas é chamado de transdutor.

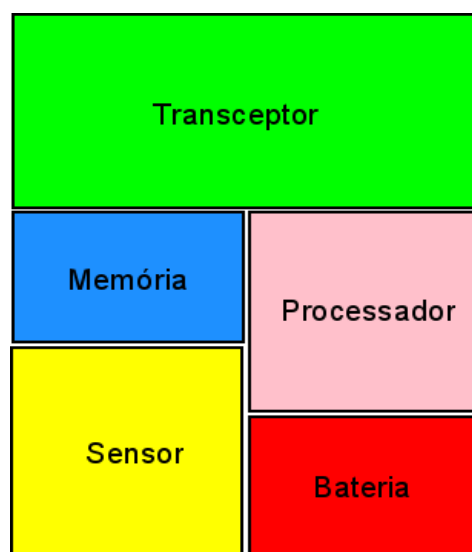


Figura 2 - Hardware básico de um nodo sensor
[LOUREIRO et al., 2003]

2.1.1.2 Nodos Gateway

Um nodo *gateway* é o responsável por conectar uma rede de sensores a outra rede. Os dados percorrem a rede de sensores até encontrar um nó *gateway* que os encaminha através de uma rede como a Internet, de forma que estes dados cheguem a um computador onde roda uma aplicação que fará uso destes dados. A figura 3 mostra um esquema de uma RSSF que conectar-se a outra rede por meio de um *gateway*, enquanto a figura 4 apresenta uma rede com um nodo sorvedouro ou *sink* e um *gateway* mostrando que são componentes distintos.

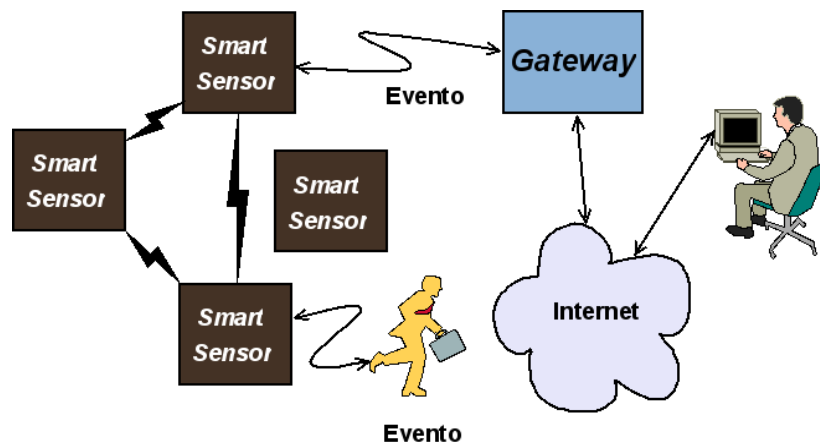


Figura 3 - Modelo genérico de uma RSSF com um gateway
[LOUREIRO et al., 2003]

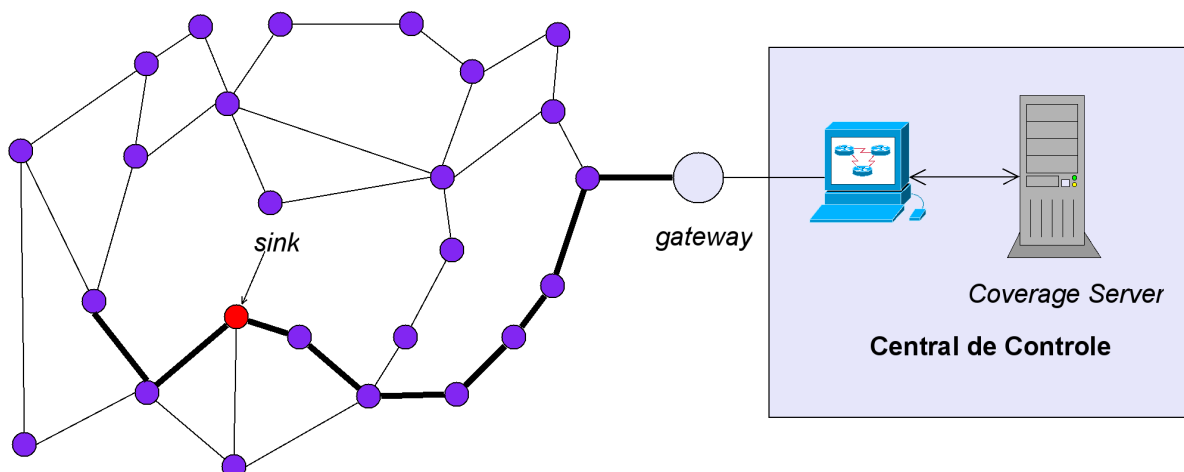


Figura 4 - Modelo de RSSF com um nó sink [LOUREIRO et al., 2003]

2.1.2 Características das Redes de Sensores Sem Fio

Uma RSSF é concebida para detectar eventos ou fenômenos, coletar e processar dados e transmitir o conteúdo sensoriado aos usuários interessados. As características básicas das RSSF são [HAENGGI, 2005]:

- Capacidade de auto-organização;
- Difusão da comunicação em pequenas escalas e roteamento *multihop*;
- Posicionamento denso e estratégico de nodos que trabalham de forma cooperativa;

- Mudança freqüente de topologia devido à falhas dos nodos;
- Limitações em energia, potência de transmissão, memória e capacidade computacional.

Estas características, principalmente as últimas três, tornam as RSSF diferente das outras redes *ad hoc* e *mesh*.

2.1.3 Protocolos de Comunicação em Redes de Sensores Sem Fio

Em uma rede de sensores sem fio, o principal papel da subcamada de acesso ao meio é reduzir ou até mesmo evitar o conflito pacotes. Algumas características destas redes demandam protocolos especializados, entre elas estão [ZHAO, 2004]:

- RSSF são sistemas colaborativos normalmente servindo uma ou um pequeno número de aplicações. Desta forma problemas de imprecisões nos sensores são menos importantes que a performance do sistema como um todo;
- Em muitas das redes de sensores, os nodos estão ociosos durante grande parte do tempo. Quando um evento é detectado por estes nodos, ocorre um aumento repentino de atividade em uma determinada área da rede, provavelmente longe de onde a informação solicitada é necessária. Em função disso, as aplicações devem estar preparadas para lidar com grandes tempos de latência;
- Processamento em rede pode melhorar consideravelmente a utilização da banda;
- Problemas de mobilidade podem ser tratados por meio de um bom protocolo de acesso ao meio;
- Os principais problemas com os quais se depara nas RSSF são de eficiência de energia, escalabilidade e robustez. No entanto, o maior objetivo dos protocolos deve ser prolongar a vida da rede.

Segundo [ZHAO, 2004], protocolos de controle de acesso ao meio foram desenvolvidos para redes sem fio seja para transmissão de voz seja para transmissão de dados. Entre estes protocolos estão: TDMA e CDMA utilizados em redes de celulares,

CSMA para Ethernet e WLAN (Wireless Local Area Network). Ainda segundo ZHAO, a mais importante publicação sobre protocolos para controle de acesso ao meio para RSSF partiu da Universidade da Califórnia, Los Angeles. A publicação cita o protocolo S-MAC. Outra abordagem importante parte do padrão IEEE 802.15.4 para aplicações embarcadas e de baixo custo.

2.1.3.1 O Protocolo S-MAC

O principal objetivo do protocolo S-MAC é reduzir a perda de energia causada por escuta em tempo de ociosidade, colisões, escuta desnecessária e controle de *overhead*. Este protocolo possui quatro componentes principais: escuta e dorme periodicamente, componente para evitar colisões, componente para evitar escuta desnecessária e mensagens transitórias.

Escuta e dorme periodicamente exerce o papel de reduzir o consumo de energia durante os longos tempos de ociosidade, enquanto não são detectados eventos. Para reduzir a latência e controle de *overhead*, o protocolo S-MAC busca gerenciar tempos de ociosidade entre os nós vizinhos por meio da troca do horário de repouso entre estes nós.

O componente de controle de colisões trabalha de forma que, se o nó não consegue acesso ao meio este entra em modo ocioso e volta ao modo ativo quando o receptor esta livre e escutando novamente. O nó sabe quanto tempo pode ficar em repouso cada pacote possui a informação de quanto tempo sua transmissão irá durar. Desta forma o componente para evitar escutas desnecessárias, coloca o nó em repouso enquanto seus vizinhos trocam mensagens entre si.

Mensagens são tratadas como unidades de dados trocadas entre sensores, mensagens grandes são divididas em pacotes, para isso, o meio fica reservado para a transmissão completa desta mensagem. Assim são reduzidos custos com controle e a latência das mensagens. Isso não significa que uma pequena mensagem precisará aguardar o fim da transmissão de uma grande mensagem.

Em função dos longos tempos de inatividade dos nodos em RSSF, o protocolo S-MAC atualmente apresenta-se mais vantajoso que o 802.11.

2.1.3.2 Padrão IEEE 802.15.4 e ZigBee

O padrão IEEE 802.15.4 define protocolos para a camada física e de controle de acesso ao meio para controle e monitoração remota. ZigBee é um consorcio com o objetivo de promover o padrão 802.15.4. ZigBee assegura a interoperabilidade por meio da definição de camadas elevadas e interfaces de aplicação.

O baixo custo e baixo consumo de energia do padrão 802.15.4 permitem longa vida de baterias, características necessárias a aplicações de baixo custo para monitoramento.

Diferente do S-MAC, o padrão 802.15.4 trabalha a eficiência de energia nas camadas física e de controle de acesso ao meio. Espera-se que o ciclo ativo de comunicação no padrão 802.15.4 seja em torno de 1%, resultando em um consumo de energia muito baixo.

2.1.4 TinyOS e nesC

TinyOS é um sistema operacional de código aberto desenvolvido para ser utilizado em redes de sensores sem fio. Entre suas características está a arquitetura baseada em componente que permite implementações com pouca codificação. Outra característica interessante do TinyOS é ser um sistema operacional orientado a eventos. Esta característica permite um melhor gerenciamento da energia e mais flexibilidade no agendamento de ações. A biblioteca de componentes do TinyOS já possui protocolos de comunicação [TINYOS, 2008].

De forma a tornar-se ainda mais simplificado, TinyOS não possui sistema de arquivos, suporta apenas alocação estática de memória, implementa um modelo tarefas simples e provê uma abstração mínima do dispositivo e da rede [ZHAO, 2004].

2.1.4.1 nesC

nesC é uma extensão do C concebida para suportar e refletir a arquitetura do TinyOS [ZHAO, 2004].

A linguagem nesC foi desenvolvida para atender às necessidades especiais de redes utilizando sistemas embarcados (*Networked Embedded Systems*), é uma

linguagem que incorpora a execução orientada a eventos, modelo de concorrência flexível arquitetura de aplicação orientada a componente [GAY, 2003].

3 Ondas Sonoras, Sistemas de Sonorização e Sensores para Medição de Som

3.1 Ondas Sonoras

Ondas sonoras são vibrações longitudinais do meio material – sólido, líquido ou gasoso – que são transmitidas através dele com uma velocidade determinada precisamente pelas propriedades mecânicas do próprio meio. Em geral, as ondas sonoras são geradas por alguma espécie de vibração mecânica causada por atrito, batidas ou outras formas de dissipação de energia em arranjos mecânicos, naturais ou artificiais [REFERENCE, 1978].

Neste trabalho serão consideradas ondas geradas por sistemas de sonorização e propagadas no ar.

3.1.1 Efeitos da temperatura sobre as ondas sonoras

Segundo [REFERENCE, 1978], o ar pode ser considerado um gás perfeito, logo podem ser aplicadas ao mesmo a lei geral dos gases. Assim, a velocidade do som aumenta aproximadamente 0,6 m/s para cada grau centígrado que a temperatura aumente. Este resultado foi confirmado por Greely, em 1890 no Ártico, para um intervalo de temperatura de -10° a -45°C .

Ondas sonoras produzidas no ar por fontes que têm velocidade maior que a do próprio som, de nenhuma forma se assemelham às ondas sonoras ordinárias. A importância deste tipo de observação levou a uma nova unidade de medida para quando um objeto se desloca no ar com velocidade do som, esta unidade é chamada número de *Mach*. Este número é adimensional e é a razão da velocidade do objeto para a velocidade do som. Portanto, para a velocidade supersônica o número de *Mach* é maior que 1, enquanto que para as velocidades usuais, subsônicas, menor que 1.

3.1.2 Efeitos da Umidade e outros fatores

O efeito da umidade na atmosfera é reduzir a densidade desta última abaixo

daquela que teria o ar seco à mesma temperatura e alterar ligeiramente a razão dos calores específicos, assim a velocidade do som aumenta ligeiramente à medida que a umidade aumenta. Outro fator que influencia na velocidade do som é sua intensidade, quando muito elevadas como em explosões [FERENCE, 1978].

As ondas geradas por sistemas de sonorização podem ser consideradas de baixa intensidade se comparadas a explosões, sendo assim, para este trabalho, pode-se desconsiderar este tipo de efeito sobre as ondas sonoras.

3.2 Sistemas de Sonorização

Sistemas de sonorização são sistemas utilizados para amplificar e distribuir o som gerado por uma determinada fonte que pode ser um palestrante, um grupo musical ou outra fonte qualquer [CYSNE, 1997]. Estes sistemas são compostos por equipamentos eletrônicos com funções distintas [MACHADO, 2004]. Abaixo é apresentada uma lista simplificada de equipamentos encontrados em um sistema de sonorização. A figura 5 traz alguns exemplos de equipamentos.



Figura 5 - Mesas e caixas de som, amplificadores e microfones

Microfones: são transdutores que transformam ondas sonoras em sinais elétricos que através de cabos são transferidos a um console de mixagem (mesa de som). Os

microfones são utilizados por palestrantes, cantores e para captação de alguns instrumentos musicais, principalmente instrumentos de percussão. Uma abordagem mais detalhada sobre microfones é encontrada no item 3.2.1.

- Instrumentos elétricos: são instrumentos musicais que não necessitam de microfones para sua captação. Em sua concepção estão preparados para disponibilizar o sinal elétrico semelhante ao gerado pelos microfones. São exemplos de instrumentos elétricos os teclados e as guitarras.
- Console de *mixagem* ou mesa de som: Equipamento eletrônico com entradas e saídas de sinais elétricos. As entradas recebem os sinais dos microfones e instrumentos elétricos, e as saídas são normalmente conectadas aos amplificadores que serão vistos a seguir. Um console de mixagem permite que o técnico de áudio equalize cada instrumento/microfone de forma independente, buscando uma melhor reprodução do som original nas caixas de som do sistema. Após equalizado cada componente o técnico faz a *mixagem*, que consiste de misturar o som dos canais de entrada buscando reproduzir o produto gerado pelo grupo musical / palestrante da forma mais fiel possível. Consoles de mixagem podem ter variados tamanhos e portarem tecnologias das mais simples às mais avançadas.
- Amplificadores: são equipamentos eletrônicos que têm a função de amplificar o sinal recebido do console de *mixagem* e enviar o sinal amplificado às caixas de som. Amplificadores são encontrados no mercado em diversas potências e finalidades e são em sua maioria equipamentos preparados para instalação em bastidores de 19”;
- Caixas de som: compartimentos de madeira rigorosamente projetados para melhor aproveitamento do som gerado pelos auto-falantes. Sua finalidade é transformar o sinal elétrico recebido dos amplificadores em ondas sonoras. As caixas de som são encontradas exercendo duas funções básicas em sistemas de sonorização: a) retorno – utilizada para que o músico ou palestrante possa ouvir seu instrumento ou sua voz; b) PA (sigla para *Public Addressed*) são as caixas de som que estão direcionadas para o público.

3.2.1 Tipos de Microfones

Microfones são diferenciados em função de sua diretividade e sua forma de captação ou tipo do transdutor [MACHADO, 2004].

Classificação quanto à diretividade:

- Ominidirecional – Capta em todos os sentidos em relação ao seu eixo (360° de ângulo de cobertura). Utilizados geralmente como microfones de lapela e encontrado também em decibelímetros. São geralmente imunes a ruídos provocados pelo vento;
- Cardióides – Diferente do ominidirecional, este apresenta áreas de maior e menor sensibilidade. Seu ângulo de cobertura é de 180°. Microfones cardióides são muito utilizados por cantores;
- Supercardióides e Hiper cardióides – Possuem ângulos de cobertura ainda mais restritivos que os cardióides, 152° e 140° respectivamente. Estes tipos de microfones são comumente utilizados por cantores e para captura do som de instrumentos percussivos como baterias;
- Bidirecionais – Possuem captação em dois lados, muito utilizados em entrevistas;
- Shotgun – O mais restritivo quanto à diretividade entre os microfones, utilizado para captação à distância.

Classificação quanto ao tipo de transdutor:

- Dinâmicos – Sua cápsula é formada por uma membrana presa a uma bobina que fica suspensa envolta por um ímã. Ondas sonoras vibram a membrana que transmite o movimento à bobina. Este movimento gera na bobina um campo magnético variável que por sua vez gera tensão nos terminais da bobina.
- Condensadores – Também conhecidos por microfones capacitivos têm seu princípio de funcionamento baseado na alteração da capacitância de seu circuito interno. Estes microfones dependem de fonte de energia externa. Microfones de

eletreto são um tipo mais simples de microfone capacitivo que pode ser alimentado por pequenas fontes como pilhas de 1,5V.

3.2 Medição de Pressão Sonora (decibéis)

Este tópico foi escrito baseado no Apêndice 1 – Decibéis de [CYSNE, 1997].

O bel (nome dado em homenagem a Alexander Grahan Bell) é a unidade básica definida como logaritmo de base 10 de uma relação de duas potências elétricas.

$$L1 \text{ em bels} = \log(P1/P2)$$

Onde P1 e P2 são potências elétricas. No entanto, para um elevado número de aplicações o bel mostrou-se uma unidade muito grande, tornando-se em alguns momentos inconveniente sua notação. Passou a ser utilizado então o decibel (dB), que corresponde a um décimo do bel. Desta forma:

$$L1 \text{ em dB} = 10 \log(P1/P2)$$

Em engenharia de áudio, para medição de pressão sonora é utilizada como referência para cálculo da relação $20\mu\text{N/m}^2$ que é o limiar da audição humana, ou seja, é a pressão à partir da qual começamos a ouvir sons. No lugar do multiplicador 10 é utilizado o multiplicador 20.

Por exemplo, para que se possa definir qual a pressão sonora em dB que corresponda à pressão dinâmica de 2N/m^2 , deve-se calcular $Lp = 20 \log(2/0,00002)$.

$$Lp = 20 \log(100000) = 100\text{dB}$$

No ano de 1956 Robinson e Dadson levantaram as curvas chamadas “contornos de mesma audibilidade”. Cada uma destas curvas representa a pressão sonoras que diferentes freqüências devem ter para parecerem tão audíveis quanto 1kHz. Estas curvas foram traçadas para base na freqüência de 1kHz em diferentes pressões sonoras e são apresentadas na Figura 6.

Com base nas curvas levantadas por Robinson e Dadson, os medidores de pressão sonora utilizam curvas de ponderação de forma que os instrumentos de medida

se comportem aproximadamente como o ouvido humano. As curvas são A, B e C, apresentadas na Figura 7 e as medições efetuadas por instrumentos que utilizam estas curvas de ponderação devem ser identificadas por dB(A), dB(B) e dB(C) respectivamente.

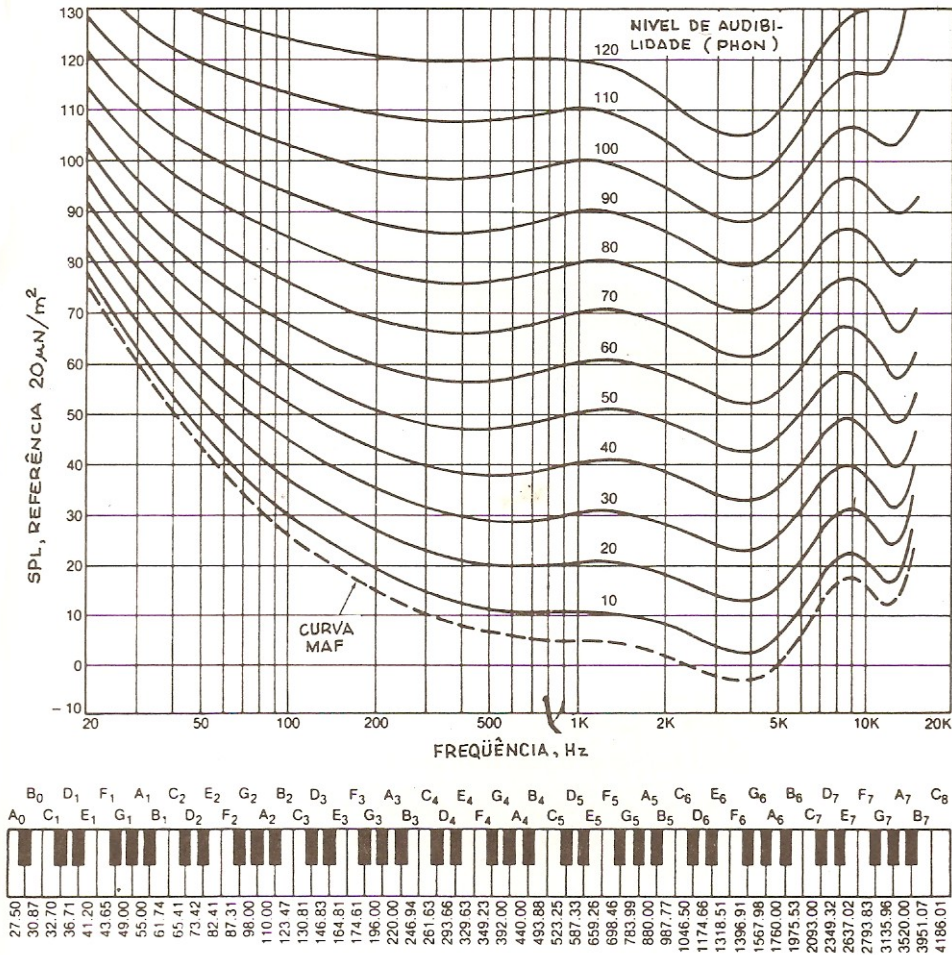


Figura 6: Contornos de Mesma Audibilidade [CYSNE, 1997]

Com o objetivo de permitir ao leitor referência de níveis de ruído, a Tabela 1

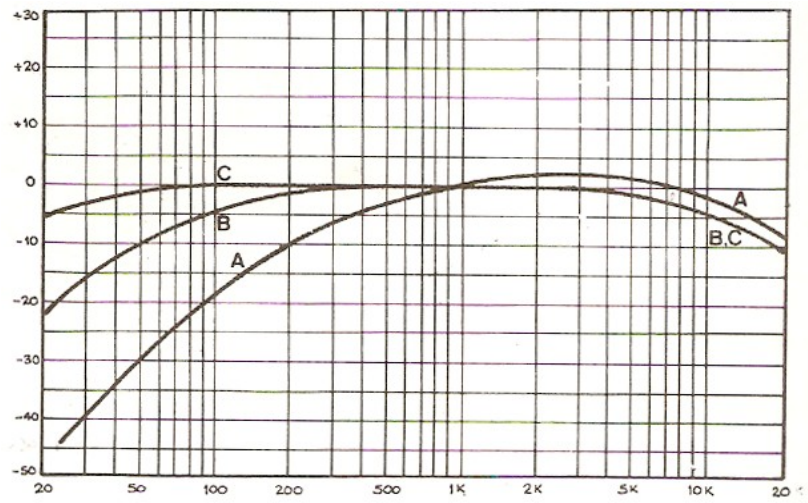


Figura 7: Curvas de Compensação A, B e C [CYSNE, 1997]

apresenta uma lista de atividades e o nível de ruído a ela associado.

Tabela 1: Relação entre atividade e nível de ruído

dB(A) ref. 20 μ N/m ²	ATIVIDADE
0	Limiar da audição para jovens, entre 1 e 4kHz
7	Câmara anecóica bem construída
10	Sala a prova de som
20	Estúdio musical para cinema
25	Estúdio musical de rádio e TV
30	Estúdio genérico para gravação de voz
32	Suspiro suave a 4 metros
35	Biblioteca com boa acústica
40	Níveis mínimos em áreas residenciais de pequenas cidades
50	Pequeno escritório
50	Média em residências
60	Escritório de contabilidade
60	Áreas residenciais urbanas barulhentas
63	Aquecedor a gás a 1,5 metro
70	Trem de carga a 30 metros
70	Supermercado
70	Áreas residenciais muito barulhentas
80	Tráfego pesado de veículos
82	Furadeira pneumática a 15 metros
90	Salão com máquinas impressoras
92	Metrô a 5 metros
100	Imediações de fornalha elétrica
105	Serra profissional
110	Máquina rebitadeira
120	Trovão
121	Decolagem de jato a 80 metros

3.3.1 Medições Aplicáveis a Sistemas de Sonorização

A monitoração de um sistema de som é, normalmente, efetuada apenas de forma empírica pelo técnico que, usando de sua experiência, deve ser capaz de distinguir pela

intensidade e equalização ideal para o ambiente.

Atualmente existem equipamentos utilizados para efetuar medições em sistemas de sonorização. Um exemplo de medição é a análise em tempo real da resposta do sistema (RTA – *Real Time Analyzer*). Um equipamento pode efetuar uma comparação entre o sinal enviado aos amplificadores e som reproduzido nas caixas de som, traçando assim uma curva de resposta do sistema.

Um exemplo de equipamento com esta função é o Ultracurve PRO DEQ2496 da marca Behringer, mostrado na figura 8.



Figura 8 - Ultra Curve Behringer

Um equipamento mais comum utilizado para medições de som é o decibelímetro. Instrumento muito utilizado por técnicos de áudio, técnicos de segurança no trabalho e estudiosos da área de poluição ambiental. A figura 9.a mostra o modelo ITDEC-460 e 9.b o modelo ITDEC-200, digital e analógico respectivamente.



Modelo ITDEC-200

Modelo ITDEC-460

Figura 9 - Decibelímetros

3.4 Efeitos do ruído sobre o organismo humano

Segundo [DO CARMO, 1999], frente à exposição a ruídos em geral, as orelhas são dotadas de mecanismos protetores que alteram a sensibilidade auditiva durante e após a estimulação acústica. O primeiro mecanismo protetor é o mascaramento, que se caracteriza pela atenuação da percepção de um som quando há um ruído de intensidade que encubra o primeiro. O segundo é a adaptação auditiva, que reduz a sensibilidade auditiva durante a apresentação de estímulo sonoro intenso e duradouro. O terceiro diz respeito à fadiga auditiva, que ocorre após cessados os estímulos, também conhecido por mudança temporária.

Para fins de referência, a tabela 2 apresenta os números fornecidos pelo Ministério do Trabalho por meio da Norma Regulamentadora nº 15 que indica níveis de ruído e a máxima exposição diária do indivíduo.

Tabela 2: Máxima exposição diária permissível

Nível de Ruído dB(A)	Máxima Exposição Diária Permissível
85	8 horas
86	7 horas
87	6 horas
88	5 horas
89	4 horas e 30 minutos
90	4 horas
91	3 horas e 30 minutos
92	3 horas
93	2 horas e 40 minutos
94	2 horas e 15 minutos
95	2 horas
96	1 hora e 45 minutos
98	1 hora e 15 minutos
100	1 hora
102	45 minutos
104	35 minutos
105	30 minutos
106	25 minutos
108	20 minutos
110	15 minutos
112	10 minutos
114	8 minutos
115	7 minutos

3.5 Sensores para medição de som

Entre os módulos sensores estudados até o momento, dois modelos verificados como adequado para o trabalho são o MTS300 e o MTS310 por possuírem um microfone para capturar os ruídos do ambiente monitorado. A diferença entre os dois modelos é a presença de sensores de magnetismo e o acelerômetro no MTS310. A figura 10 mostra uma placa de sensores MTS310 e uma plataforma MPR400CB.

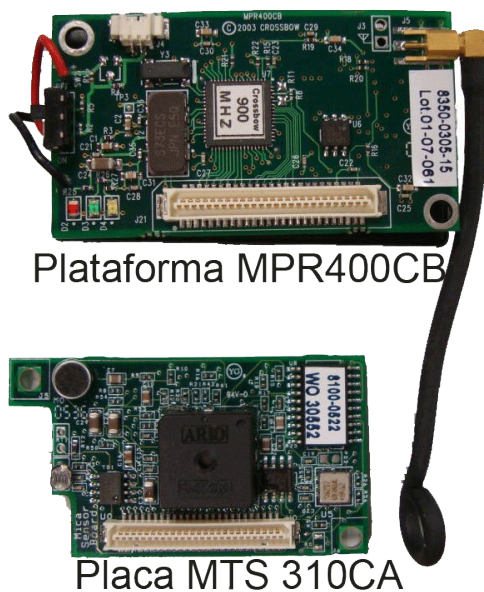


Figura 10: Plataforma e Placa de Sensores

MTS300 e 310 são placas de sensores que devem ser utilizadas em conjunto com a plataforma Mica2 MPR400, ambas da marca CrossBow.

Outra placa de sensores que poderia ser utilizada no projeto é a SBT80 da marca EasySen que deve ser utilizada em conjunto com a plataforma TelosB TPR2400CA da marca CrossBow. A figura 11 apresenta uma placa SBT 80 conectada a uma plataforma TelosB.



Figura 11: SBT80 conectada a uma plataforma TelosB

4 SPLDM - Monitoração de Pressão Sonora utilizando Redes de Sensores Sem Fio

O objetivo de um sistema de sonorização é amplificar o som gerado seja por um grupo musical, seja por um palestrante, e distribuir este produto amplificado de maneira estratégica no ambiente de maneira que a platéia possa ouvir de forma mais homogênea possível independente de sua localização [CYSNE, 1997]. O técnico de áudio que opera este sistema começa seu trabalho na passagem de som muitas vezes horas antes do show e prossegue enquanto durar o evento. Em muitas situações está sob sua responsabilidade o cuidado com a distribuição da pressão sonora no ambiente e, conforme cita [DO CARMO, 1999], este técnico sofrerá mudanças em sua sensibilidade auditiva no decorrer da apresentação, gerando assim diferenças em sua percepção auditiva entre o início e o fim da apresentação. Um bom exemplo de ambiente no qual ocorre a dinâmica citada é o das igrejas.

Desde o final dos anos 80 o ambiente das igrejas vem trabalhando de forma crescente com grupos musicais dos mais variados estilos. O público destas igrejas entretanto continua distribuído entre as mais diferentes as faixas etárias, que por sua vez exprimem diferentes opiniões sobre estilos musicais mas principalmente sobre o popular “volume” do som, aqui tratado como pressão sonora.

O técnico de áudio tem a responsabilidade de manter o som nítido, agradável e dentro de níveis saudáveis aos ouvintes. Para esta tarefa o profissional faz uso do decibelímetro, equipamento já citado no capítulo 3, deslocando-se no ambiente para efetuar medições em pontos estratégicos do local. Uma das dificuldades encontradas nesta tarefa é que as leituras estão deslocadas no tempo, impedindo uma comparação direta entre as mesmas.

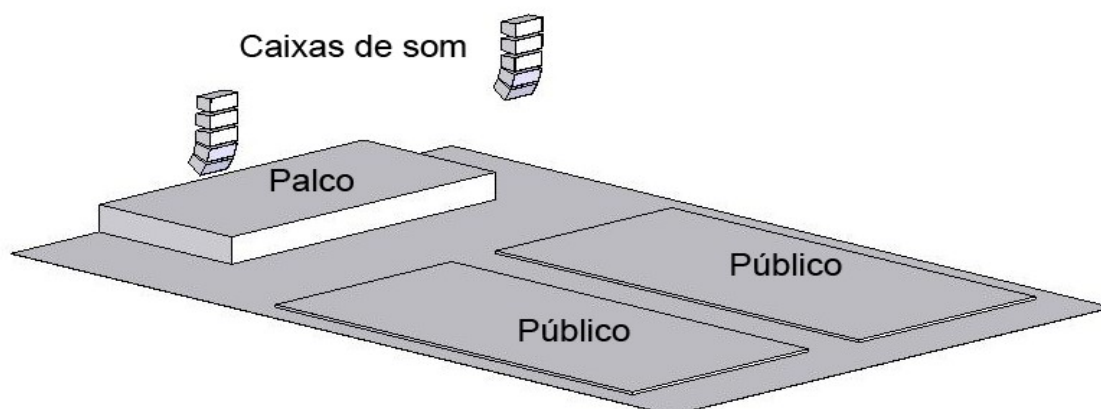


Figura 12 - Ambiente para sonorização

A figura 12 apresenta um exemplo de um ambiente com caixas de som dispostas de forma que as superiores projetam o som para o público do fundo da sala e as inferiores para o público próximo ao palco.

4.1 SPLDM – Sound Pressure Level Distributed Meter

Este trabalho apresenta uma proposta de monitoração de um ambiente sonorizado utilizando para tal uma rede de sensores sem fio, sensores estes equipados com microfones de forma a apresentar ao técnico de áudio uma leitura da distribuição da pressão sonora no ambiente.

Com o uso de sensores distribuídos no ambiente, é possível monitorar a pressão sonora em alguns pontos, mostrando os resultados ao técnico que por sua vez não precisará deslocar-se para obter um parecer da distribuição da pressão sonora citada.

A figura 13 mostra de forma simplificada a arquitetura física do sistema de monitoramento.

O SPLDM consiste de sensores distribuídos no ambiente monitorado que enviam informações a um nodo central, este por sua vez envia os dados recebidos à porta serial do computador no qual estará em execução a aplicação responsável por tratar os dados recebidos e apresentá-los ao usuário.

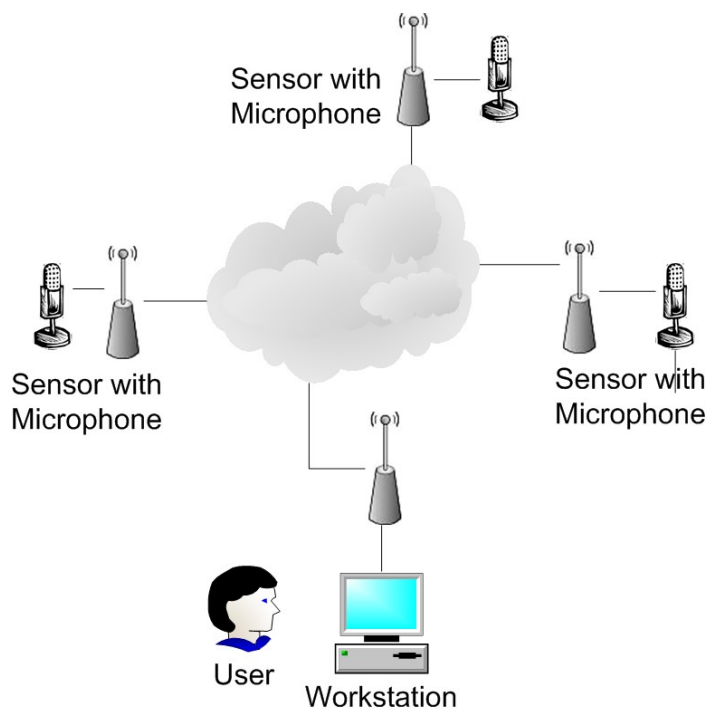


Figura 13: Rede de sensores para monitorar um ambiente sonorizado

A figura 14 apresenta o diagrama lógico da solução proposta e a seguir uma explicação sobre cada módulo da figura.

- *Environment* – o ambiente monitorado pelos sensores (ex.: igreja, salão de festas, casa de shows);
- *Sensor 1..n* – cada nodo sensor distribuído no ambiente;
- *Sensor 0* – nodo base, interface da rede de sensores com a plataforma de aquisição de dados;

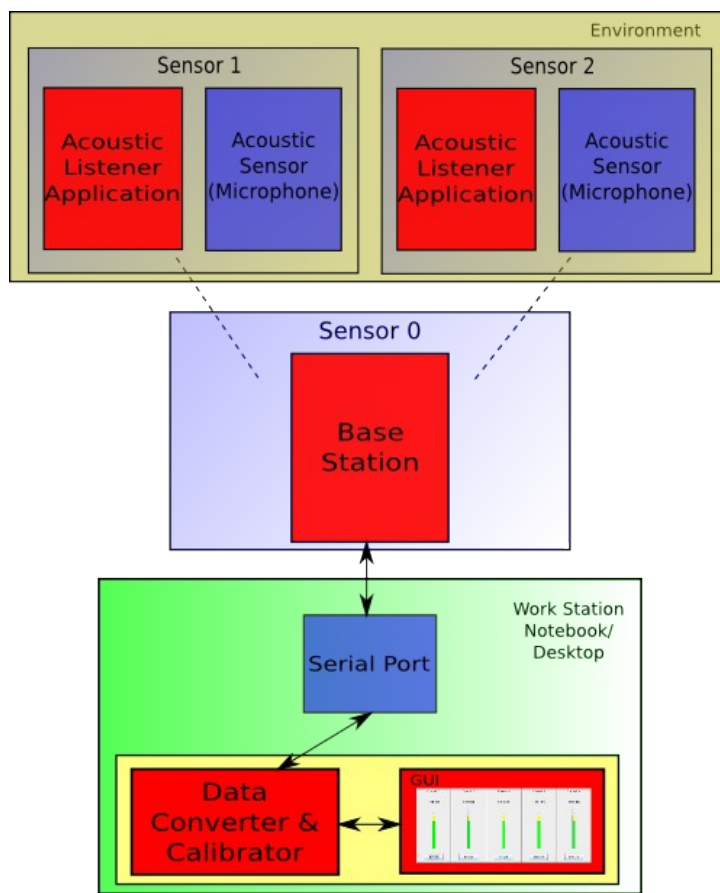


Figura 14: Diagrama Lógico - SLPDM

- Acoustic Listener Application* – Software desenvolvido na linguagem nesC que tem por função capturar dados da placa de sensores, mais especificamente do sensor acústico (microfone). Os dados são capturados em blocos de 10 amostras com o uso da interface `ReadStream` que é implementada pelo componente `MicStream`. `ReadStream` deve ser inicializada recebendo um *buffer* e a quantidade de amostras que serão inseridas neste por meio da função `postBuffer(val_t* buf, uint16_t count)`. A função `read(uint32_t usPeriod)` deve ser invocada informando o intervalo entre as amostras para o *buffer*. A função para postagem do *buffer* pode ser invocada várias vezes, recebendo referências para diferentes *buffers*, assim, para cada *buffer* cheio o evento `bufferDone(error_t result, val_t* buf, uint16_t count)` é disparado, bem como o evento `readDone(error_t result, uint32_t usActualPeriod)` quando todos os *buffers* da fila já foram utilizados. Neste trabalho utilizado apenas 1 *buffer*. Para cada coleta de dados, os mesmos são enviados para o nodo base que será explicado abaixo. *Acoustic Listener Application* foi desenvolvido com base na

aplicação *Oscilloscope* existente no ambiente de desenvolvimento do TinyOS 2.

- *Base Station* – Conforme descrito em `/tinios-2.x/apps/BaseStation/README.txt`, *Base Station* atua como uma ponte entre a porta serial e os rádios (nodos sensores) da rede. Esta aplicação é encontrada no ambiente de desenvolvimento do TinyOS 2 entre as aplicações de exemplo.
- *Serial Port* – Este módulo representa a porta serial do computador. Também pode ser observado como uma porta USB utilizando um cabo adaptador para serial.
- *Data Converter & Calibrator* – Módulo desenvolvido em Java baseado na aplicação *Listen* também encontrada entre os exemplos do TinyOS2 (`tinios-2.x/support/sdk/java/net/tinios/tools`). Por meio de JNI comunica-se com a porta serial recebendo os dados dos sensores em formato de bytes. Separa os pacotes por ID do sensor para que os dados de cada sensor seja mostrado em um diferente canal da interface gráfica. Visto que são recebidas 10 leituras em cada pacote, o módulo faz uma média entre os valores recebidos antes de enviar o dado à interface gráfica. Faz-se necessário tornar mais rigoroso o tratamento das leituras (mais detalhes em trabalhos futuros).
- GUI – Interface gráfica para interação com o usuário. Mostra a leitura da pressão sonora efetuada por cada nodo sensor. Deve ser possível nesta interface verificar a a pressão sonora nos diferentes pontos do ambiente onde encontra-se cada sensor.

As contribuições do trabalho proposto encontram-se na aplicação em execução nos sensores (Acoustic Listener Application) e na aplicação de tratamento e apresentação dos dados. Os demais módulos podem ser encontrados no conjunto de ferramentas para desenvolvimento de aplicações do TinyOS 2.

Para uma melhor compreensão da interação entre os módulos é apresentado na Figura 15 o diagrama um diagrama de seqüência do sistema.

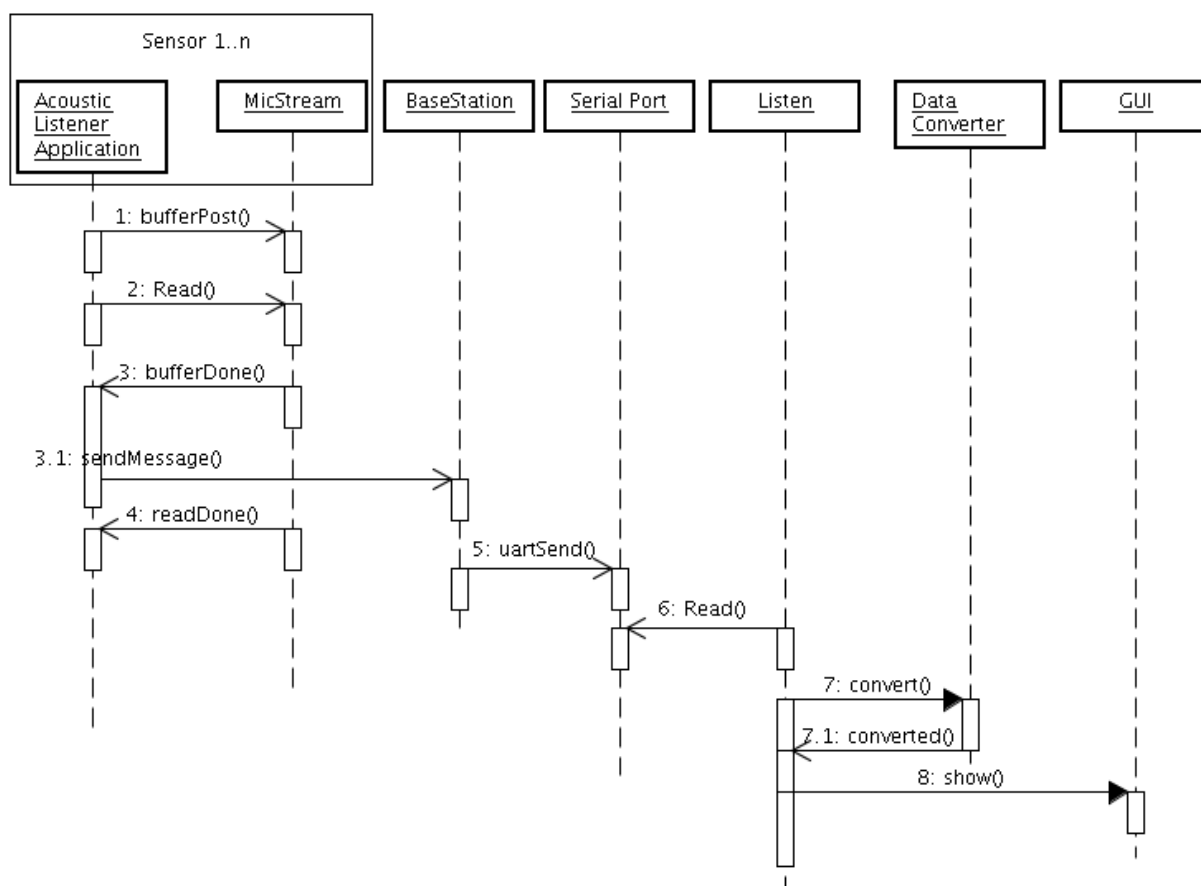


Figura 15: Diagrama de Seqüência

4.2 Trabalhos Relacionados

Entre os trabalhos consultados no estudo das RSSF, o que chamou mais atenção e permitiu vislumbrar a viabilidade do desenvolvimento deste o foi *Real-Time Acoustic Monitoring Using Wireless Sensor Motes* (Monitoração acústica em tempo real utilizando sensor sem fio Motes). O trabalho foi desenvolvido por Visar Berisha, Homin Kwon e Andreas Spanias do Departamento de Engenharia Elétrica da Universidade do Arizona. O trabalho consiste de um sistema de segurança para monitorar o ambiente buscando descobrir a característica do invasor por meio da voz deste [BERISHA et al,2006].

Outro trabalho que despertou interesse foi o de [YUNES, 2007] que consiste de um sistema de monitoração acústica utilizado para estudar o ambiente de sobrevivência de uma determinada raça de sapos.

No período de pesquisa para desenvolvimento da aplicação deparou-se com o

trabalho desenvolvido por Christian Hermann, que consiste de um *middleware* para prover informações de redes de sensores sem fio a várias aplicações ao mesmo tempo. Em geral, cada solução apresentada utilizando redes de sensores sem fio faz seu próprio acesso à rede de forma particular, Hermann propõe um compartilhamento de dados de uma mesma rede a várias aplicações[HERMANN, 2007]. Dentro da tese de Hermann o ponto principal de interesse foi o do teste com os sensores acústicos (microfones) que se assemelha em muito com a proposta aqui apresentada.

Tabela 3: Correlação dos Trabalhos Relacionados

	Plataforma	Sensor Utilizado	Tempo Real
Berisha, V.	MICA2	Acústico	Sim
Yunes, Y.	AOpen MSN	Acústico	Não
Hermann, C.	MICAz	Acústico, Luz, Temperatura, Magnetismo e Aceleração	Não

5 Ambiente, Implementação e Resultados Experimentais

5.1 Ambiente de Desenvolvimento

Nesta sessão é apresentado o ambiente para os experimentos bem como detalhes relativos à implementação. Em adição são apresentados os resultados experimentais relativos aos estudos de casos.

5.1.1 Ferramentas

Para desenvolvimento das aplicações em Java e nesC foi utilizada a plataforma Eclipse com o plugin TinyOS. Esta plataforma foi escolhida em função da experiência profissional e acadêmica já adquirida. Para compilação das aplicações em Java foi adicionado ao *classpath* do Eclipse a biblioteca *tinyos.jar*, que acompanha a instalação do TinyOS 2. O plugin TinyOS até sua utilização apresentava-se preparado apenas para a versão 1.x, auxiliando apenas e na coloração da sintaxe dos fontes nesC para TinyOS 2.x, ainda assim de forma incompleta.

5.1.3 Hardware

Os recursos utilizado para a execução dos experimentos foram:

- 4 Plataformas MPR400 CB marca Crossbow;
- 3 Placas de Sensores MTS 310CA marca Crossbow;
- 1 Interface Serial MIB 510 marca Crossbow;
- 1 Cabo Adaptador USB-Serial marca ST Lab;
- 1 Decibelímetro Analógico Radio Shack ajustado para utilizar a curva de compensação A e velocidade de resposta rápida;
- Notebook Acer 5630-6091 – Aplicação Java e envio de sinal ao FW 330;
- Aparelho de Som Philips FW 330 exercendo função de amplificador;

- 1 Caixa de Som Philips FB 330 conectada ao FW 330;
- Faixa 28 (1kHz) do CD Áudio Teste – Sinais e Música para Avaliação Eletroacústica, elaborado por Sólon do Valle e comercializado pela editora Áudio Música & Tecnologia;

A definição da faixa de 1kHz para os testes baseou-se nos seguintes aspectos;

- Simplificação do escopo;
- É uma frequência audível;
- É a frequência normalmente utilizada no ajuste de sinais de áudio;
- Pode-se considerar que esta faixa não é afetada pelas curvas de compensação de medição citadas no módulo Decibéis;
- Cada mensagem enviada pelos nodos possui 10 leituras efetuadas com intervalo de 0,1ms (mili segundos), totalizando 1ms de amostra, equivalente a 1 ciclo da senóide de 1kHz;

5.2 Implementação

5.2.1 Nodos

5.2.1.1 Nodos Sensores

Os nodos sensores são os elementos que efetivamente capturam informações do ambiente. A aplicação desenvolvida para captura dos sinais sonoros teve como base a aplicação *Oscilloscope* encontrada no tutorial do TinyOS2. Da aplicação tomada como referência dois aspectos principais foram mantidos:

- Estrutura da mensagem;
- Comunicação direta entre nodos sensores e nodo base.

As principais modificações efetuadas foram:

- Sensor utilizado;
- Interface de acesso ao sensor, passando da interface *Read* para *ReadStream*;
- Envio de mensagens apenas ao nodo base, evitando a escuta desnecessária por parte dos demais nodos;
- Modificação no fluxo de funcionamento para não ocorrerem paralelamente envio de mensagens e coleta de dados do microfone;

5.2.1.2 Nodo Base

O nodo base conecta a rede de sensores ao computador que vai processar as informações desta rede. Para funcionamento do nodo base foi utilizada a aplicação *BaseStation* encontrada no tutorial do TinyOS2.

5.2.2 Aplicação Java

A aplicação Java, conforme já citado foi desenvolvida utilizando a plataforma Eclipse e faz uso da biblioteca *tinyos.jar* que acompanha a instalação do TinyOS. Seu módulo de comunicação com a porta serial é baseado na classe *Listen* que pode ser encontrada nos fontes da biblioteca *tinyos.jar*.

5.3 Resultados Experimentais

Os resultados experimentais demandaram algumas tentativas de conversão dos dados sem sucesso. Foi mudada a estratégia de forma a observar o comportamento dos dados e buscar uma maneira de conversão. Para tal foi definida uma tarefa exclusiva de captura de dados antes da calibração, o Microsoft Excel foi utilizado como ferramenta de auxílio para determinação da função de cada sensor. Determinadas as funções estas foram incorporadas à aplicação Java, de forma a transformar os valores dos registradores dos sensores em valores equivalente em dB (decibéis). Estes passos serão detalhados a seguir nos itens 5.3.1 e 5.3.2.

5.3.1 Coleta de Dados para Calibração

Para o reconhecimento do comportamento dos registradores dos sensores e posterior calibração, foram coletados dados com as pressões de 60dB(A), 70dB(A), 80dB(A), 90dB(A) e 100dB(A).

A Figura 16 apresenta os dados coletados com a emissão de pressões sonoras de 60dB(A), 70dB(A), 80dB(A), 90dB(A) e 100dB(A), e a tabela 4 apresenta a média e o desvio padrão para cada um dos sensores e para cada uma das coletas. Nos gráficos é possível observar o crescimento dos valores com o crescimento da pressão sonora.

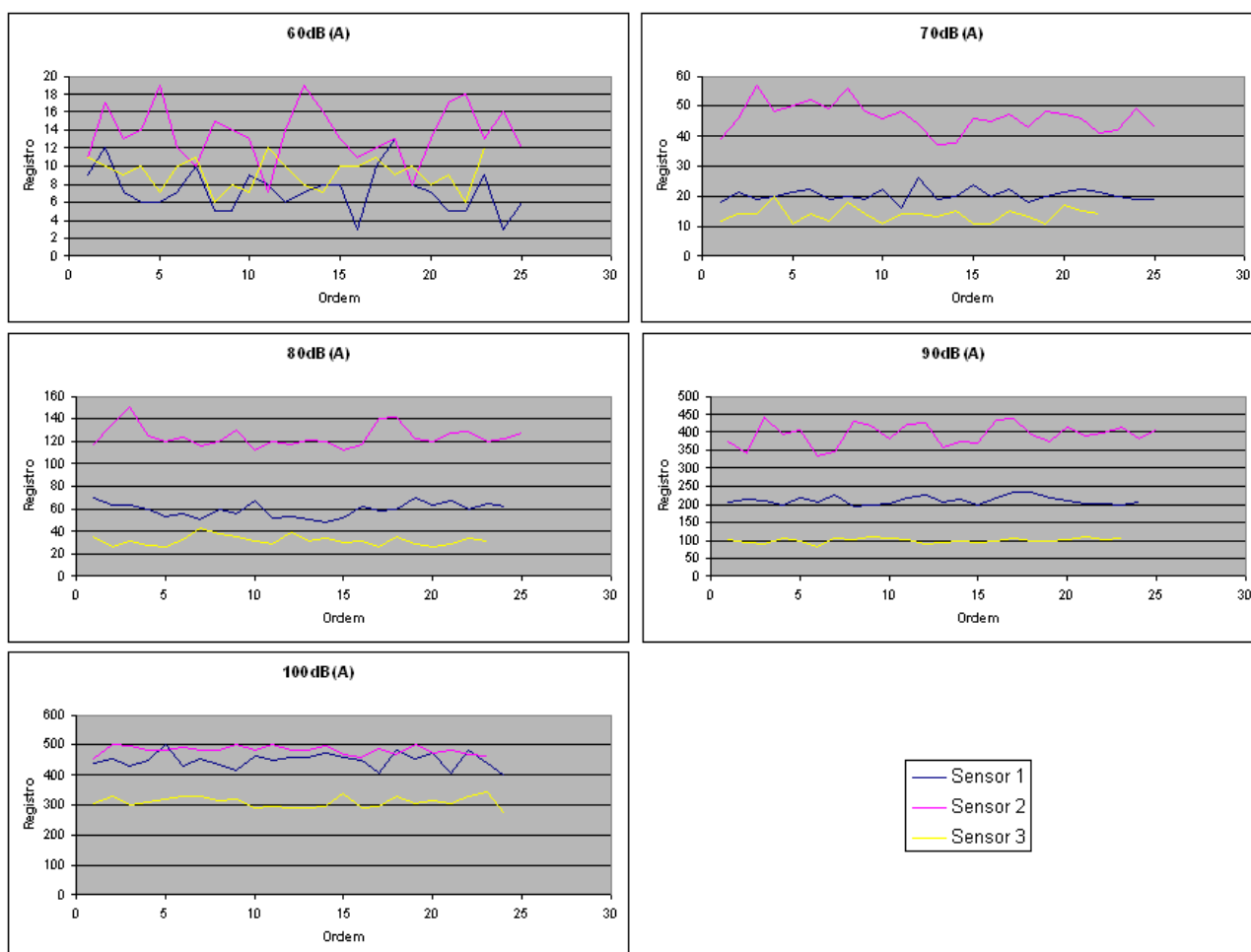


Figura 16: Registros dos sensores para pressões sonoras de 60, 70, 80, 90 e 100dB(A)

Em todas as medições foi possível observar que o sensor 2 apresenta maior sensibilidade, ou seja, seus registros sempre estavam acima dos demais. Esta característica fez com que na medição de 100dB(A) o sensor 2 estivesse em saturação. Visto que no momento dos experimentos não se encontravam disponíveis protetores auriculares, não foram efetuadas medições com pressões acima de 100dB(A).

Tabela 4: Média e desvio padrão das medições dos sensores

Pressão(db(A))	Sensor 1		Sensor 2		Sensor 3	
	Média	D. Padrão	Média	D. Padrão	Média	D. Padrão
60	7,28	2,46	13,60	3,07	9,17	1,77
70	20,32	2,04	46,20	4,86	13,77	2,37
80	59,13	6,44	124,20	9,41	31,61	4,53
90	209,83	10,92	394,36	30,96	99,30	6,69
100	448,04	25,40	481,83	14,20	309,42	18,56

Obtenção da função dos sensores

Utilizando-se das médias contidas na Tabela 4, foi possível a montagem do gráfico da Figura 17. Neste gráfico é possível observar que todos os sensores apresentaram uma curva com tendência logarítmica. Para permitir que a aplicação SPLDM apresenta-se em sua interface valores em dB, foi necessário conhecer a função de cada um dos sensores. Para esta tarefa foi utilizado como ferramenta o Microsoft Excel.

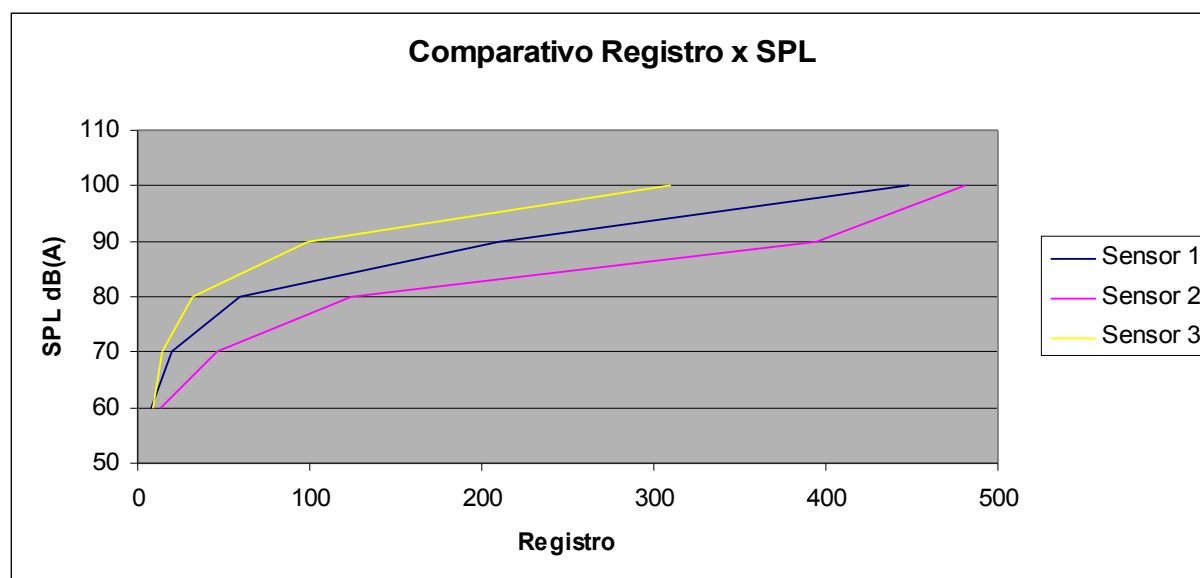


Figura 17: Comparativo entre pressão sonora e o registro dos sensores 1, 2 e 3

O Microsoft Excel é capaz de traçar uma linha de tendência muito próxima a uma linha já existente em um gráfico e desta nova linha informar sua função e sua aderência à linha original. Este recurso da ferramenta está disponível quando com o cursor aponta-se para uma das linhas do gráfico e pressiona-se o botão direito do mouse. Este passo é demonstrado na Figura 18.

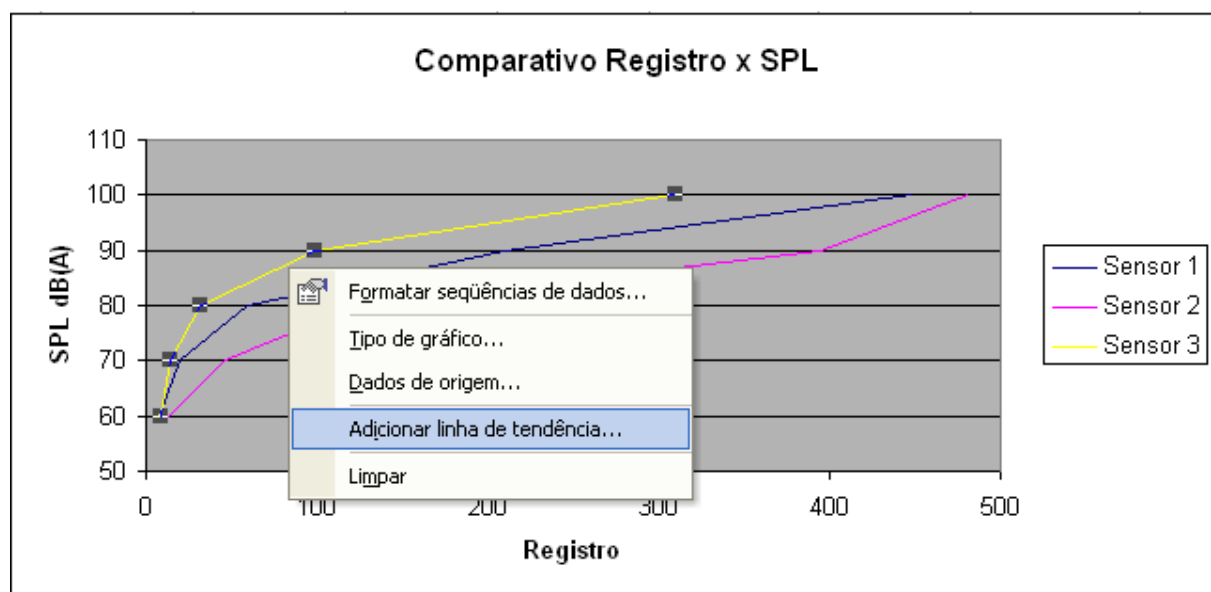


Figura 18: Geração de Linha de Tendência com o Microsoft Excel

Selecionada a opção “Adicionar linha de tendência”, será apresentada a tela da Figura 19a, neste momento deve-se selecionar o padrão de linha que se deseja, para o caso apresentado foi selecionado a linha Logarítmica. Na aba Opções, apresentada na Figura 19b, é possível marcar as opções “Exibir equação no gráfico” e “Exibir o valor de R-quadrado no gráfico”, que irão inserir a função da nova linha e sua aderência à linha original no gráfico respectivamente.

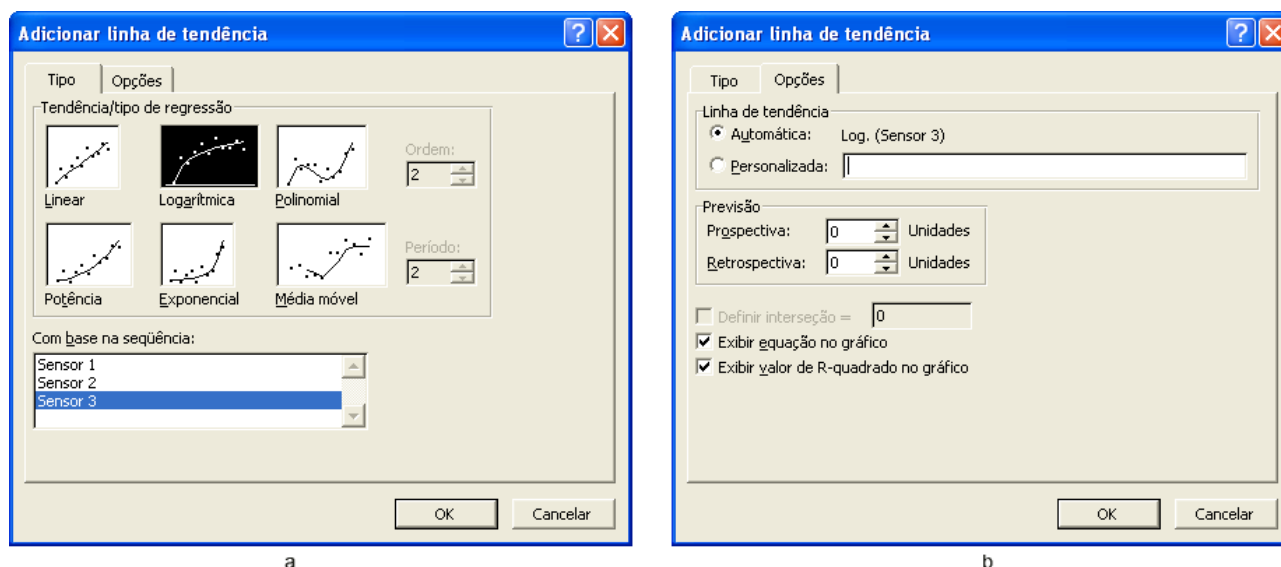


Figura 19: Opções da Geração de Linha de Tendência

Após executados estes passos para as linhas dos 3 sensores, foram encontradas linhas de tendência muito próximas às linhas originais, permitindo a utilização da função destas novas linhas na aplicação SPLDM para transformar os valores dos registros dos

sensores em decibéis (dB). Os resultados da utilização da ferramenta são apresentados na Figura 20 e as funções e o R^2 de cada uma das linhas na Tabela 5.

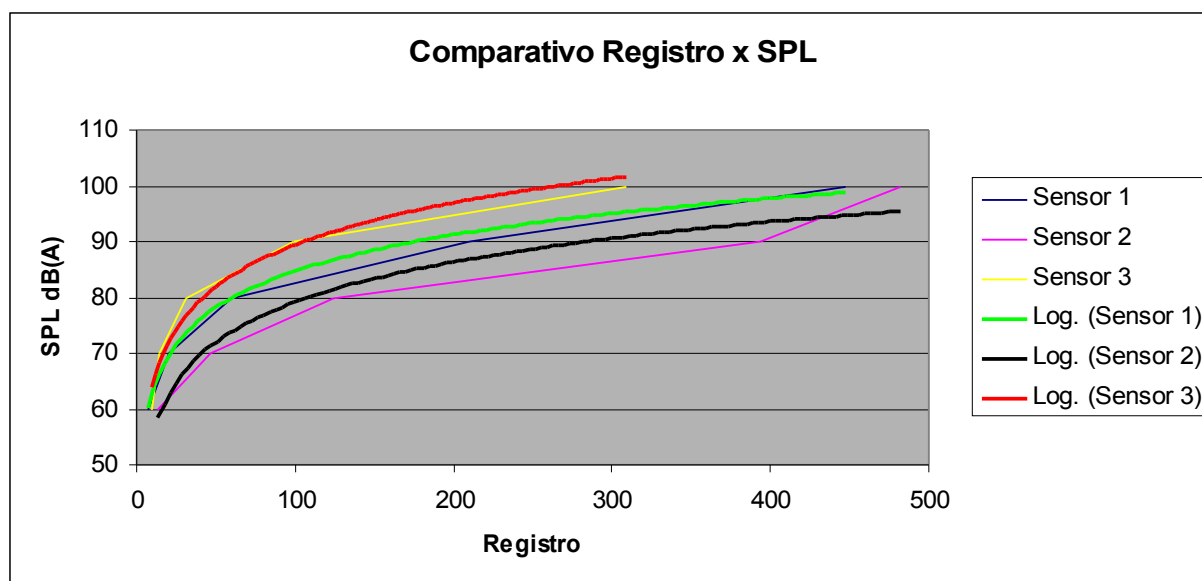


Figura 20: Linhas originais e de tendência dos sensores 1, 2 e 3

Tabela 5: Função e R^2 das Linhas de Tendência

	Sensor 1	Sensor 2	Sensor 3
Função	$y = 9,4157\text{Ln}(x) + 41,344$	$y = 10,371\text{Ln}(x) + 31,425$	$y = 10,772\text{Ln}(x) + 39,873$
R^2	0,9956	0,9623	0,9708

5.3.2 Medições

Após conhecidas as funções das linhas de tendência, estas passaram a fazer parte da aplicação SPLDM mais especificamente no módulo de calibração. Desta forma, a cada mensagem recebida por um dos nodos sensores, calcula-se a média das 10 leituras por ele efetuadas com intervalo de 0,1ms, esta média é aplicada à respectiva função de acordo com o sensor e assim obtém-se o valor em decibéis que é apresentado na interface para o usuário. A Figura 21 apresenta a tela do SPLDM durante a medição de pressões sonoras de 60dB(A), 70dB(A), 80dB(A), 90dB(A) e 100dB(A). A aplicação SPLDM direcionou os valores dos sensores já calibrados para os canais da interface conforme a Tabela 6.

Comparando a imagens da Figura 21d e 21e, que com um aumento de 10dB(A) o SPLDM no canal 3 registrou um aumento de apenas 2dB, isso se deu pois como já citado nas medições para calibração o sensor 2 em quando submetido a pressões de 100dB(A) entrava em saturação.

Tabela 6: Interligação dos sensores à interface

Sensor 1	Sensor 2	Sensor 3
Canal 2	Canal 3	Canal 4

Os sensores 1 e 3 (canal 2 e 4) apresentaram comportamento bastante linear com poucas taxas de erro de medição.

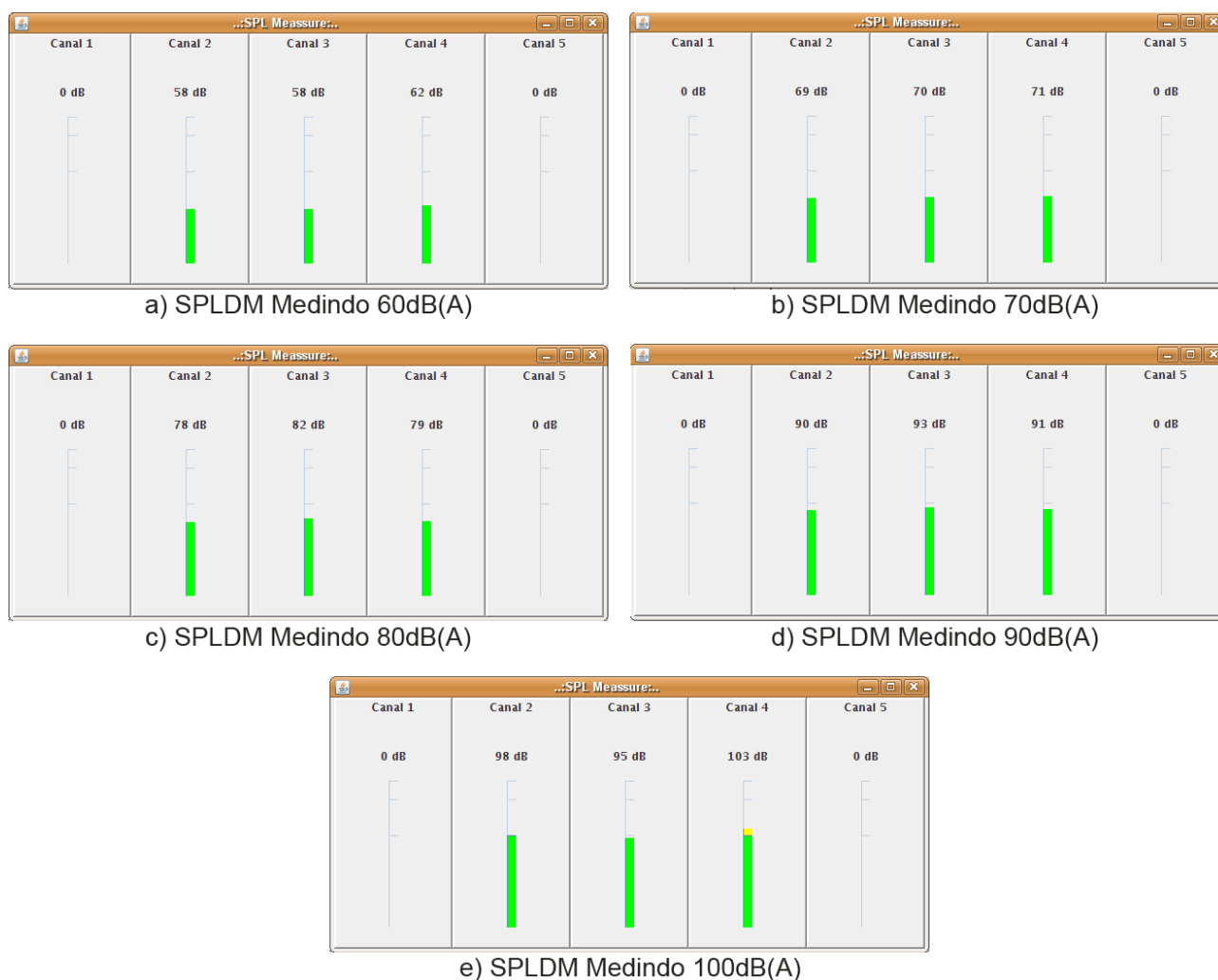


Figura 21: Telas do SPLDM com medições de 60dB(A), 70dB(A), 80dB(A), 90dB(A) e 100dB(A)

Na Figura 22 podem ser observados os valores medidos e esperados. Mais uma vez é possível observar a mudança no comportamento do sensor 2 no extremo superior da faixa de medição.

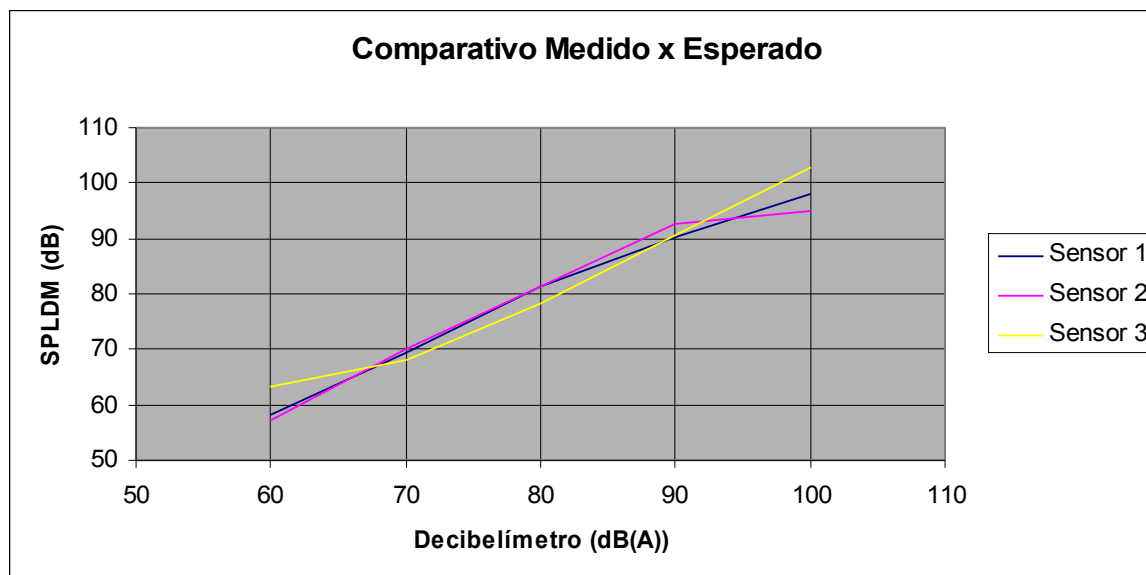


Figura 22: Comparativo Medido x Esperado

Uma opção que poderia melhorar a precisão da conversão dos dados dos sensores em decibéis, seria a segmentação em faixas de valores onde em cada uma destas faixas seria utilizada uma função diferente, buscando uma maior aproximação dos valores esperados.

Cabe reforçar que nas medições de prova como nas leituras para calibração o sinal sonoro utilizado foi uma senóide de 1kHz.

6 Conclusões e Trabalhos Futuros

6.1 Conclusões

O desenvolvimento deste trabalho de conclusão permitiu aprofundamento sobre linguagens de programação, alguns fundamentos matemáticos, parte da tecnologia das RSSF, fundamentos de física e áudio.

Os experimentos utilizando redes de sensores sem fio para medição de pressão sonora permitiram otimismo para a continuidade do trabalho de pesquisa. Foi possível conhecer algumas das limitações do hardware dos equipamentos utilizados bem com sua repetibilidade nas leituras.

As medições de comprovação no item 5.3.2 demonstraram certa precisão na leitura por meio da aplicação SPLDM bem como a limitação dos sensores quando o nível de pressão sonora é elevado (acima de 90dB(A)).

Pode-se concluir que as redes de sensores sem fio se mostraram ferramentas interessantes para medição e monitoração de pressão sonora ainda que considerando suas limitações. Para que seja possível conceber um produto com este fim para utilização junto a sistemas de sonorização profissional, seria necessário circuitos elaborados de forma a apresentar um comportamento um pouco mais flexível. Um exemplo desta flexibilidade seria a possibilidade de ajuste da sensibilidade do sensor acústico no próprio hardware. A monitoração de frequências mais altas também pode ser de complexidade mais elevada, no entanto não foram efetuados testes para esta verificação.

6.2 Trabalhos Futuros

- Ampliação da faixa de análise (20Hz – 20kHz);
- Modificações na aplicação de forma a permitir calibração dos sensores pelo usuário;
- Busca de hardware que permita medições acima de 100dB;
- Elaboração de algoritmo que permita uma calibração mais precisa dos nodos sensores por meio da segmentação em faixas de medição;
- Interação com sistemas de som para realimentação e controle da pressão sonora;

- Nova abordagem para medição não apenas de pressão sonora mas também análise de espectro.

Referências

BERISHA, V; KWON, H; SPANIAS, A. Real-Time Acoustic Monitoring Using Wireless Sensor Network – Arizona State Universit - ISCAS 2006

CYSNE, L. F. O.; Áudio, Engenharia e Sistemas 4ª Edição– Rio de Janeiro, Ed. H Sheldon - 1997

CROSSBOW TECHNOLOGY – <http://www.xbow.com> – Último acesso em maio/2008

DO CARMO, L. I. C.; Efeitos do ruído ambiental no organismo humano e suas manifestações auditivas – CEFAC Centro de Especialização em Fonoaudiologia Clínica, 1999

ERENCE JR, M; LEMON, H. B.; STEPHENSON, R. J. [trad. e adapt. de] GOLDEMBERG, J. Curso de Física Ondas (luz e som). São Paulo – Edgard Blücher, 1978

GAY, D. et al The nesC Language: A Holistic Approach to Networked Embedded Systems – San Diego – Califórnia – EUA, 2003

HAENGGI, M. Opportunities and Challenges in Wireless Sensor Networks – Univesidade de Notre Dame, 2005

HERMANN, C. Middleware for Supporting Multiple Applications in Wireless Sensor Networks – Universidade de Tecnologia de Dresden - 2007

LOUREIRO, A. A. F. et al Redes de Sensores sem Fio. XXI Simpósio Brasileiro de Redes de Computadores, 2003

MACHADO, R. M.; Som ao Vivo 2ª Edição– Conceitos e aplicações básicas em sonorização – Rio de Janeiro, Ed. H. Sheldon - 2004

TINYOS – <http://www.tinyos.net> – Último acesso em junho/2008

YUNES, Y. Acoustic Signal Representation for Environmental Surveillance Monitoring (ESM) – Universidade de Porto Rico - 2007

ZHAO, F.; GUIBAS, L. Wireless Sensor Network: An information processing approach, Morgan Kaufmann Publications, 2004, ISBN: 1-55860-914-8

ZHAO, F; LIU, J.; LIU, J.; GUIBAS, L.; REICH, J Collaborative Signal and Information Processing: An Information Directed Approach , 2003

Anexos

Código fonte da aplicação Java

```

package spl;

import spl.gui.MainPanel;

public class Main {

    private static Listen listen;
    private static MainPanel mainPanel;

    /**
     * @param args
     */
    public static void main(String[] args) {

        try {
            mainPanel = new MainPanel();
            mainPanel.setVisible(true);
            listen = new Listen("serial@/dev/ttyUSB0:mica2",
mainPanel.getPannels());
            listen.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

/*
 * Desenvolvido com base no Listen existente no TinyOS 2
 */

package spl;

import java.io.IOException;
import java.util.ArrayList;

import spl.gui.PanelVU;

import net.tinyos.packet.BuildSource;
import net.tinyos.packet.PacketSource;
import net.tinyos.util.PrintStreamMessenger;

public class Listen extends Thread {

    private PacketSource reader;

    private ArrayList<PanelVU> paineis;
    private int count = 0 ;

    public Listen(String pPortaConexao, ArrayList<PanelVU> paineis) throws
IOException {
        this.paineis = paineis;
        reader = BuildSource.makePacketSource(pPortaConexao);
        if (reader == null) {

```

```

        System.err.println("Invalid packet source (check your MOTECOM
environment variable)");
        System.exit(2);
    }
}

public void run() {
    byte[] packet = null;

    try {
        reader.open(PrintStreamMessenger.err);
        System.out.println("count\tsensor\tpressao");
        for (;;) {
            packet = reader.readPacket();
            int[] intDados = arrayByteToInt(packet);
            alimentaPaineis(intDados);
        }
    } catch (IOException e) {
        System.err.println("Error on " + reader.getName() + ": " + e);
    }
}

private void alimentaPaineis(int[] dados) {
    try {
        int sensorId = (dados[12] << 8) + dados[13];
        int posicaoLeitura = 16;
        // System.out.println("sensorId: " + sensorId);

        int[] leituras = new int[10];
        for (int i = 0; i < leituras.length; i++) {
            leituras[i] = dados[posicaoLeitura] << 8;
            posicaoLeitura++;
            leituras[i] += dados[posicaoLeitura];
            posicaoLeitura++;
        }

        int registroMedio = calculaLeituraMedia(leituras);
//        System.out.println(registroMedio);
        int pressao = CalculadorPressao.getPressao(registroMedio,
sensorId);

        System.out.println(count + "\t" + sensorId + "\t" + pressao);
        count++;
        paineis.get(sensorId).setBarValue(pressao);
    } catch (Exception e) {
        // Pode acontecer de chegarem mensagens com erro. Aqui é onde
vai refletir isso...
        e.printStackTrace();
    }
}

private int calculaLeituraMedia(int[] leituras) {
    int i;
    int sampleSum;
    int noiseSum;
    int sampleMid;

    sampleSum = 0;
    noiseSum = 0;
    sampleMid = 0;
}

```



```

        for (i = 0; i < leituras.length; ++i) {
            sampleSum += leituras[i];
        }
        sampleMid = sampleSum / (leituras.length);
        //System.out.println("sampleMid: " + sampleMid);

        for (i = 0; i < leituras.length; ++i) {
            noiseSum += Math.abs(leituras[i] - sampleMid);
        }

        int noiseMid = noiseSum/leituras.length;
        return noiseMid;
    }

    private int[] arrayByteToInt(byte[] bytes) {
        int[] inteiros = new int[bytes.length];
        for (int i = 0; i < bytes.length; i++) {
            inteiros[i] = (int) bytes[i] & 0xFF;
        }
        return inteiros;
    }
}

package spl;

public class CalculadorPressao {

    private static double fator = 0;

    private static double offSet = 0;

    /*
     * Valores calculados com base nos testes efetuados e através das curvas
     de tendência
     *
     * Sensor 1
     *  $y = 9,4157\ln(x) + 41,344$ 
     *  $R2 = 0,9956$ 
     *
     * Sensor 2
     *  $y = 10,371\ln(x) + 31,425$ 
     *  $R2 = 0,9623$ 
     *
     * Sensor 3
     *  $y = 10,772\ln(x) + 39,873$ 
     *  $R2 = 0,9708$ 
     */

    public static int getPressao(int registro, int sensorId) {
        int pressao = 0;
        switch (sensorId) {
            case 1:
                fator = 9.4157;
                offSet = 41.344;
                break;

            case 2:
                fator = 10.371;
                offSet = 31.425;
        }
    }
}

```

```

        break;

    case 3:
        fator = 10.772;
        offSet = 39.873;
        break;
    default:
        break;
}

pressao = (int) (fator*Math.log(registro) + offSet);
return pressao;
}
}

package spl;

public class IConstants {

    public static int MAX_LEVEL_MEASURE = 140;

    private IConstants() {

    }

}

package spl.gui;

import java.awt.Container;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.util.ArrayList;

import javax.swing.JFrame;

public class MainPanel extends JFrame {

    private static final long serialVersionUID = 1L;
    private ArrayList<PanelVU> panels;

    private PanelVU panelVU0;
    private PanelVU panelVU1;
    private PanelVU panelVU2;
    private PanelVU panelVU3;
    private PanelVU panelVU4;

    public MainPanel() {
        super("...SPL Measure...");
        init();
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void init() {
        panelVU0 = new PanelVU("Canal 1");
        panelVU1 = new PanelVU("Canal 2");
        panelVU2 = new PanelVU("Canal 3");
        panelVU3 = new PanelVU("Canal 4");
        panelVU4 = new PanelVU("Canal 5");

        panels = new ArrayList<PanelVU>();
        panels.add(panelVU0);
    }
}

```

```

panels.add(panelVU1);
panels.add(panelVU2);
panels.add(panelVU3);
panels.add(panelVU4);

Container container = this.getContentPane();

GridBagLayout layout = new GridBagLayout();
container.setLayout(layout);

GridBagConstraints c = new GridBagConstraints();
c.gridx = 0;
c.gridy = 0;
c.gridheight = 1;
c.gridwidth = 1;

c.fill = GridBagConstraints.BOTH;
c.anchor = GridBagConstraints.CENTER;

c.weightx = 1;
c.weighty = 1;

layout.setConstraints(panelVU0, c);
container.add(panelVU0);

c.gridx = 1;
layout.setConstraints(panelVU1, c);
container.add(panelVU1);

c.gridx = 2;
layout.setConstraints(panelVU2, c);
container.add(panelVU2);

c.gridx = 3;
layout.setConstraints(panelVU3, c);
container.add(panelVU3);

c.gridx = 4;
layout.setConstraints(panelVU4, c);
container.add(panelVU4);

this.setBounds(300, 300, 650, 300);
}

public ArrayList<PanelVU> getPanels() {
    return panels;
}
}

package spl.gui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.BorderFactory;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;

```

```

import javax.swing.SwingUtilities;

import spl.IConstants;

public class PanelVU extends JPanel {

    private static final long serialVersionUID = 1L;
    private JProgressBar barGreen;
    private JProgressBar barYellow;
    private JProgressBar barRed;
    private JLabel jLabelName;
    private JLabel jLabelLevel;

    private String name;

    public PanelVU(String name){
        super();
        this.name = name;
        init();
    }

    private void init(){
        this.setBorder(BorderFactory.createRaisedBevelBorder());

        GridBagLayout layout = new GridBagLayout();
        setLayout(layout);

        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        c.gridheight = 1;
        c.gridwidth = 1;

        c.fill = GridBagConstraints.NONE;
        c.anchor = GridBagConstraints.NORTH;

        c.weightx = 1;
        c.weighty = 1;

        jLabelName = new JLabel(name);
        jLabelLevel = new JLabel("0 dB");

        layout.setConstraints(jLabelName, c);
        this.add(jLabelName);

        c.anchor = GridBagConstraints.SOUTH;
        c.gridy++;
        layout.setConstraints(jLabelLevel, c);
        this.add(jLabelLevel);

        c.anchor = GridBagConstraints.SOUTH;
        c.gridy++;
        barRed = new JProgressBar(JProgressBar.VERTICAL);
        barRed.setPreferredSize(new Dimension(10, 20));
        barRed.setMinimum((int) (IConstants.MAX_LEVEL_MEASURE * 0.9));
        barRed.setMaximum(IConstants.MAX_LEVEL_MEASURE);
        barRed.setBorder(BorderFactory.createEmptyBorder());
        layout.setConstraints(barRed, c);
        barRed.setForeground(Color.RED);
        this.add(barRed);
    }
}

```

```

        c.anchor = GridBagConstraints.CENTER;
        c.fill = GridBagConstraints.VERTICAL;
        c.gridy++;
        barYellow = new JProgressBar(JProgressBar.VERTICAL);
        barYellow.setPreferredSize(new Dimension(10, 20));
        barYellow.setMinimum((int) (IConstants.MAX_LEVEL_MEASURE * 0.7));
        barYellow.setMaximum((int) (IConstants.MAX_LEVEL_MEASURE * 0.9));
        barYellow.setBorder(BorderFactory.createEmptyBorder());
        layout.setConstraints(barYellow, c);
        barYellow.setForeground(Color.YELLOW);
        this.add(barYellow);

        c.fill = GridBagConstraints.NONE;
        c.anchor = GridBagConstraints.NORTH;
        c.gridy++;
        barGreen = new JProgressBar(JProgressBar.VERTICAL);
        barGreen.setPreferredSize(new Dimension(10, 100));
        barGreen.setMinimum(0);
        barGreen.setMaximum((int) (IConstants.MAX_LEVEL_MEASURE * 0.7));
        barGreen.setBorder(BorderFactory.createEmptyBorder());
        layout.setConstraints(barGreen, c);
        barGreen.setForeground(Color.GREEN);
        this.add(barGreen);
    }

    public void setBarValue(final int valor){
        SwingUtilities.invokeLater(
            new Runnable() {
                public void run() {
                    barGreen.setValue(valor);
                    barYellow.setValue(valor);
                    barRed.setValue(valor);
                    jLabelLevel.setText(valor + " dB");
                }
            }
        );
    }
}

```

Código Fonte da Aplicação nesC

```

/*
 * Baseado na aplicação Oscilloscope do TinyOS 2
 *
 * @author Luiz Pieri
 */

#ifndef OSCILLOSCOPE_H
#define OSCILLOSCOPE_H

enum {
    /* Number of readings per message. If you increase this, you may have to
       increase the message_t size. */
    NREADINGS = 10,

    /* Default sampling period. */

```

```

DEFAULT_INTERVAL = 1,

AM_OSCILLOSCOPE = 0x93
};

typedef nx_struct oscilloscope {
    nx_uint16_t version; /* Version of the interval. */
    nx_uint16_t interval; /* Sampling period. */
    nx_uint16_t id; /* Mote id of sending mote. */
    nx_uint16_t count; /* The readings are samples count * NREADINGS onwards */
    nx_uint16_t readings[NREADINGS];
} oscilloscope_t;

#endif

/*
 * Baseado na aplicação Oscilloscope do TinyOS 2
 *
 * @author Luiz Pieri
 */

/**
 * Baseado na aplicação Oscilloscope do TinyOS 2
 *
 * @author Luiz Rogério De Pieri lrpieri@inf.ufsc.br
 */
configuration OscilloscopeAppC { }
implementation
{
    components OscilloscopeC, MainC, ActiveMessageC, LedsC,
        new TimerMilliC(), new MicStreamC() as Sensor,
        new AMSenderC(AM_OSCILLOSCOPE), new AMReceiverC(AM_OSCILLOSCOPE),
        new TimerMilliC() as TimerDelay,
        SounderC;

    OscilloscopeC.Boot -> MainC;
    OscilloscopeC.RadioControl -> ActiveMessageC;
    OscilloscopeC.AMSend -> AMSenderC;
    OscilloscopeC.Receive -> AMReceiverC;
    OscilloscopeC.Timer -> TimerMilliC;
    OscilloscopeC.TimerDelay -> TimerDelay;
    OscilloscopeC.ReadStream -> Sensor;
    OscilloscopeC.Leds -> LedsC;
    OscilloscopeC.Sounder -> SounderC;
    OscilloscopeC.MicSetting -> Sensor;
    // OscilloscopeC.MicController -> Sensor.MicSetting;
}

/**
 * Oscilloscope demo application. See README.txt file in this directory.
 *
 * @author Luiz Rogério De Pieri lrpieri@inf.ufsc.br
 */
#include "Timer.h"
#include "Oscilloscope.h"

module OscilloscopeC{
    uses {
        interface Boot;
        interface SplitControl as RadioControl;
    }
}

```

```

    interface AMSend;
    interface Receive;
    interface Timer<TMilli>;
    interface Timer<TMilli> as TimerDelay;
    interface ReadStream<uint16_t>;
    interface Leds;
    interface Mts300Sounder as Sounder;
    interface MicSetting;
}
}
implementation
{
    enum{
        MIC_SAMPLES = 10
    };
    message_t sendbuf;
    bool sendbusy;

    oscilloscope_t local;

    uint8_t reading; /* 0 to NREADINGS */

    uint16_t micBuffer[MIC_SAMPLES];

    bool suppress_count_change;

    // Utiliza os LEDs para reportar problemas
    void report_problem() { call Leds.led0Toggle(); }
    void report_sent() { call Leds.led1Toggle(); }
    void report_received() { call Leds.led2Toggle(); }

    event void Boot.booted() {
        local.interval = DEFAULT_INTERVAL;
        local.id = TOS_NODE_ID;
        if (call RadioControl.start() != SUCCESS){
            report_problem();
        }
        call MicSetting.muxSel(1);
        call MicSetting.gainAdjust(1);
    }

    void startTimer() {
        call Timer.startOneShot(local.interval);
        reading = 0;
    }

    void envia_mensagem(){
        if (!sendbusy && sizeof local <= call AMSend.maxPayloadLength()){
            memcpy(call AMSend.getPayload(&sendbuf), &local, sizeof local);
//            if (call AMSend.send(AM_BROADCAST_ADDR, &sendbuf, sizeof local) ==
SUCCESS)
                if (call AMSend.send(0, &sendbuf, sizeof local) == SUCCESS)//envia
apenas para o nodo base que precisa ser instalado com "install,0"
                    sendbusy = TRUE;
                }
        if (!sendbusy){
            report_problem();
        }
        reading = 0;
        if (!suppress_count_change){

```

```

        local.count++;
    }
    suppress_count_change = FALSE;
}

void faz_leitura(){
    call ReadStream.postBuffer(micBuffer, MIC_SAMPLES);
    if (call ReadStream.read(0.1) != SUCCESS){
        call Sounder.beep(100);
        report_problem();
    }
}

void controle(){
    if (reading == NREADINGS){
        envia_mensagem();
    }else{
        faz_leitura();
    }
}

event void Timer.fired() {
    controle();
}

event void TimerDelay.fired(){
    controle();
}

event void AMSend.sendDone(message_t* msg, error_t error) {
    if (error == SUCCESS){
        report_sent();
    }else{
        report_problem();
    }
}

sendbusy = FALSE;
call Timer.startOneShot(local.interval);
}

/*
 * Entra aqui quando encheu todos os buffers
 */
event void ReadStream.readDone(error_t result, uint32_t usActualPeriod) {

}

event void ReadStream.bufferDone(error_t result, uint16_t *buf, uint16_t
count) {
    if (result == SUCCESS) {
        uint8_t i;
        for(i = 0; i < count; i++){
            local.readings[i] = buf[i];
        }
        reading = NREADINGS;
    }else{
        report_problem();
    }
    controle();
}
}

```



```

event void RadioControl.startDone(error_t error) {
    controle();
}

event void RadioControl.stopDone(error_t error) {
}

event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len) {
    oscilloscope_t *omsg = payload;
    report_received();

    if (omsg->version > local.version){
        local.version = omsg->version;
        local.interval = omsg->interval;
    }
    if (omsg->count > local.count) {
        local.count = omsg->count;
        suppress_count_change = TRUE;
    }
    return msg;
}

async event error_t MicSetting.toneDetected() {
    return SUCCESS;
}
}

```

Makefile

```

SENSORBOARD=mts300
CFLAGS=-DCC1K_DEF_FREQ=916400000
COMPONENT=OscilloscopeAppC
include $(MAKERULES)

```