UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

DISPONIBILIZAÇÃO DE INFORMAÇÕES SENSORIAIS SEM FIO VIA UPnP

Por

RAFAEL MONTENEGRO RODRIGUES

Trabalho de Graduação em Redes de Computadores

Recife

Junho, 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

DISPONIBILIZAÇÃO DE INFORMAÇÕES SENSORIAIS SEM FIO VIA UPnP

Este trabalho foi apresentado ao programa de Graduação em Engenharia da Computação pelo Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Engenheiro da Computação.

Orientador: Djamel Fawzi Hadj Sadok

Recife

Junho, 2008

Assinaturas

Este Trabalho de Graduação representa o esforço do aluno **Rafael Montenegro Rodrigues**, orientado pelo professor **Djamel Fawzi Hadj Sadok**, sob o título "**Disponibilização de Informações Sensoriais Sem Fio via UPnP**". Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste trabalho de graduação.

_____

Rafael Montenegro Rodrigues

_____

Djamel Fawzi Hadj Sadok

Dedicatória

O grande responsável é Deus. Muito obrigado Senhor, por eu ter nascido em uma família tão boa, com paz, amor e saúde.

Dedico a toda minha família que deu mais que todas as condições necessárias para eu estar aqui, me formando. É com grande amor e carinho que os tenho e sempre os terei em minha vida. Adilson Rodrigues, Tânia Montenegro e Pedro Rodrigues são pessoas por quem eu tenho o maior orgulho e admiração, só tenho a agradecer a Deus por ter nascido nessa família.

Essa vitória também é de Aline Coimbra e família, que sempre me apoiaram nesses intermináveis cinco anos de pura batalha, esforço, estresse, correria, provas, projetos, estágios, mas que ao final de tudo a recompensa foi grande.

Obrigado pelo apoio, pela compreensão e pela motivação, sem vocês não teria forças para chegar até aqui.

Agradecimentos

Ao Corujas Software Development Team, praticamente uma família desde o início do curso. Por ordem alfabética: Pablo José, Renato Bibiano, Rodrigo Campos e o agregado Rodrigo Peixoto são mais que amigos que com certeza sei que posso contar para o resto da minha vida. Agradeço a turma iniciada em 2003-2, provavelmente a mais unida na história do Centro de Informática. Especialmente ao pessoal de Engenharia, que sempre sofreu junto com provas e projetos. Ralamos, mas também nos divertimos muito. Uma grande turma, amigos eternos.

A todos do GPRT, que me acolheram de forma indescritível. Em especial ao amigo Luís Eduardo e aos professores Djamel Sadok e Judith Kelner, que me orientaram e me apoiaram bastante no desenvolvimento desse projeto. Com certeza são pessoas inesquecíveis.

Agradeço ao Grupo Pet Informática, que contribuiu de todas as formas para meu crescimento pessoal e profissional. Fico feliz e lisonjeado de ter participado desse grupo tão bom e pró-ativo. Fazem parte: Ana Cecília, André Braga, Bruno Morato, Filipe Calegario, Flávia Chaves, meu grande companheiro de turma Chico Carvalho, Igor Duarte, Jerônimo Barbosa, Jesus Sanchez-Palencia, José Dihego, Rodolfo Vasconcelos, Fernando Mota, Luiz Guimarães, Maira Tavares, Vitor Maciel, além de grandes ex-petianos como Allynson Praxedes, Diana Rubia, Thiago Rodrigues, Tiago Buarque, Thiago Fernandes e Allan Diego. Meu agradecimento especial vai para o incansável e batalhador Fernando Fonseca, que sempre coordenou o grupo de maneira sólida, respeitosa, eficaz e carinhosa, além de defender arduamente os interesses do PET.

A todo o grupo Modcs, onde comecei a fazer pesquisa científica e a publicar. Em especial agradeço a Eduardo Tavares, a Sérgio Murilo, e a principalmente Paulo Maciel, que sempre me deram todas as condições necessárias para o aprendizado, tanto acadêmico quanto para a vida pessoal. Muito obrigado pessoal, por tudo, devo muito a vocês.

A Juliana Azevedo, por ter colaborado de alguma forma para a organização deste trabalho de graduação.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Equações

Resumo

O foco do trabalho em questão será desenvolver uma rede de sensores térmicos com o intuito de monitorar ambientes distintos e distribuídos. Pode-se dividir em três fases principais: montagem do hardware necessário para recepção e transmissão dos dados relativos à temperatura; criação de um aplicativo servidor que fará a comunicação com o hardware, adquirindo os dados e fornecendo-os em rede local, e finalmente a implementação de uma aplicação WEB responsável pela coleta e disponibilização dos dados através da internet.

O protocolo utilizado pelos sensores térmicos será o padrão *ZigBee*, enquanto a comunicação de dispositivos pela rede local será via UPnP. O aplicativo servidor será desenvolvido em C#, e a aplicação WEB em ASP.NET.

Palavras-chave: *ZigBee*, UPnP, C#, ASP.NET.

Abstract

The focus of the work will be the development of a thermal sensor network to track distinct and distributed environments. We can split it in three main phases: the hardware development, needed for temperature data received from the *ZigBees* module; creation of a server that will perform the communication with the hardware, acquiring data and supplying them over the network through the UPnP protocol; and finally the implementation of a WEB application responsible for the data collection and availability through the Internet.

The communication used by thermal sensors will be carried out through the *ZigBee* technology, a wireless data transmission pattern with low power consumption. The server module will be developed in C#, and the web application in ASP.NET.

Keywords: *ZigBee*, UPnP, C#, ASP.NET.

1. Introdução

A evolução tecnológica nos ramos da comunicação sem fio, da eletrônica digital e de sistemas eletromecânicos deu início a uma revolução na área de redes de sensores sem fio. Cada vez mais se deseja receber e transmitir continuamente informações através de meios de comunicação sem fio. Nesse âmbito, muitas pesquisas vêm sendo realizadas em todo o mundo, e aplicadas nas mais diversas áreas como medicina, segurança, turismo e educação.

Uma rede de sensores pode ser conceituada de diferentes formas. Uma delas é de uma rede sem fio composta por vários sensores pequenos sem mobilidade colocados em uma base *ad hoc*[1] a fim de detectar e transmitir alguma característica física ou comportamental do ambiente. A informação contida nos sensores é inserida em uma base central de dados [3]. Em uma definição mais voltada para a área de sistemas distribuídos, afirma-se que uma rede de sensores pode ser vista como uma classe particular de tais sistemas, onde as comunicações não dependem da localização topológica da rede [2]. Sob outro enfoque, pode-se definir uma rede de sensores como um conjunto de nós individuais que trabalham independentemente, mas que podem evoluir para uma rede.

A tecnologia de rede de sensores sem fio ou RSSF abrange o monitoramento, gerenciamento e controle do mundo físico, sendo um dos grandes exemplos de computação pervasiva[2]. Para muitos, as RSSFs têm potencial para se tornar tão importante quanto a internet, uma vez que tais redes expandirão a possibilidade dos usuários de interagirem e se conectarem remotamente com o mundo físico.

Diversas companhias no mundo estão investindo em RSSFs. Por exemplo, a *York International*, uma empresa do ramo de sistemas de ventilação que gerencia 60000 clientes em todo o mundo, pretende implantar em cinco anos centenas de milhares de redes de sensores nas unidades de ar-condicionado da sua clientela. Estas redes farão o monitoramento remoto de temperatura e enviarão dados automaticamente para os escritórios da *YORK*, aliviando o trabalho em campo de mais de 2000 técnicos. Isso, além de reduzir custos e aumentar a produtividade, poderá atrair mais parceiros comerciais.

Mesmo com a euforia por parte da comunidade científica, ainda existem dificuldades que impedem uma maior disseminação. O grande problema é o fornecimento remoto de energia para esses sensores, uma vez que seria um trabalho extremamente custoso a troca constante de baterias. Fontes alternativas de energia também são opções, como energia cinética[3] e energia solar. Uma outra forma é reduzir o máximo possível o consumo de energia desses dispositivos. Muitas pesquisas com esse objetivo já tiveram início, porém muito ainda precisa ser feito.

[1] A*d hoc* - Termo que designa algo que é criado ou usado para um problema específico ou imediato.
[2] Computação Pervasiva - O computador está inserido no ambiente de forma invisível para o usuário.
[3] Energia cinética – Energia do movimento.

1.1. Motivação

No passado, redes de sensores sem fios eram utilizadas somente em projetos militares. No presente, são utilizados em diversos contextos inclusive em inúmeras situações do cotidiano. Atualmente, estes sensores podem ser encontrados em muitas aplicações como monitoramento do meio-ambiente, gerenciamento de tráfego, controle residencial, etc. As principais dificuldades encontradas no desenvolvimento de dispositivos dotados de sensores são o alto custo de produção, e o elevado consumo de energia dos mesmos.

As tecnologias mais baratas e populares de transmissão de dados sem fio utilizam-se de radiofreqüência (RF). De acordo com [7], radiofreqüência é uma freqüência ou taxa de oscilação compreendida na faixa de 3 Hz a 300 GHz. Ainda segundo [7], essa faixa corresponde à freqüência dos sinais elétricos da corrente alternada usada para produzir e detectar as ondas de rádio.

Com o crescimento e difusão das redes de computadores, diversos padrões foram criados com o intuito de integrar dispositivos eletroeletrônicos, permitindo assim seu gerenciamento a partir de qualquer ponto da rede ou até mesmo através da internet. Surgiu então o *Universal Plug and Play* (UPnP), um conjunto de padrões criado com o intuito de tornar a comunicação de dispositivos mais prática e rápida, independente de sistemas operacionais e/ou linguagens de programação. O termo UPnP deriva-se do *plug-and-play*, que oferece mais flexibilidade na instalação, na configuração e na adição de periféricos a um computador.

O UPnP é uma tecnologia emergente lançada pelo Fórum UPnP [6]. Trata-se de um conjunto de protocolos de redes de computadores, que têm como principal objetivo simplificar a implementação e conectar diretamente redes residenciais e em ambientes corporativos, facilitando o compartilhamento de arquivos, a comunicação e o entretenimento.

Através do UPnP, é possível acessar dinamicamente uma rede e disponibilizar e/ou consumir serviços na mesma, tudo de forma automática. A partir dessa comunicação, pode-se estabelecer uma conexão direta entre si, configurando-se assim uma rede ponto a ponto. A tecnologia UPnP faz uso de Web *Services*, o novo padrão para divulgar e compartilhar serviços dinamicamente na Internet [27].

1.2. Objetivos

O objetivo geral desse trabalho é construir uma arquitetura para rede de sensores térmicos sem fio, e disponibilizar essas informações sensoriais em rede local e na WEB. Para isso, alguns objetivos específicos deverão ser cumpridos:

1. Definir uma arquitetura para sensoriamento remoto de temperatura baseada em tecnologias abertas.
2. Definir o protocolo de comunicação entre módulos *ZigBees*, que farão a leitura de seus respectivos Sensores de Temperatura e enviarão a resposta para o Gerenciador de Sensores (outro módulo *ZigBee*).
3. Definir um modelo esquemático que possibilite a prototipação dos módulos *ZigBees* com seus respectivos Sensores de Temperatura.
4. Fabricar as placas de circuito impresso.
5. Implementar um aplicativo servidor em C# que será responsável pela obtenção das informações dos sensores e a distribuição dessas informações através do padrão UPnP.

6. Desenvolver uma aplicação WEB que se comunicará com a aplicação servidora através do protocolo UPnP, requisitando as temperaturas de cada sensor periférico e disponibilizando-as através da internet.

1.3. Estrutura da Monografia

Este documento é composto por 5 capítulos, cada um composto da seguinte forma:

Capítulo 1 – Apresenta uma introdução a respeito de conceitos sobre redes de sensores sem fio e sobre o conjunto de protocolos de rede UPnP, descrevendo também uma motivação para o estudo do tema proposto.

Capítulo 2 – Descreve a arquitetura geral do *framework*. Também faz um estudo das tecnologias relacionadas com este trabalho de graduação, apresentando as vantagens e as desvantagens da utilização cada uma. O capítulo ainda aborda os trabalhos relacionados com esta monografia.

Capítulo 3 – Define a arquitetura geral do Sistema de Disponibilização de Informações Sensoriais via UPnP, apresentando seus módulos de hardware e software individualmente, além de mostrar como ocorre essa integração.

Capítulo 4 – O capítulo propõe uma implementação para a arquitetura já definida anteriormente, abordando tanto os módulos de hardware quanto os módulos de software. Ainda são apresentados os experimentos realizados e uma análise dos resultados, bem como as dificuldades encontradas.

Capítulo 5 - Apresenta as conclusões sobre o trabalho, enfatizando as principais contribuições e sugestões para projetos futuros.

2. Arquitetura do *Framework*

Neste capítulo será mostrado como está organizada a arquitetura geral do *framework*. Logo após, far-se-á um estudo detalhado das tecnologias relacionadas que viabilizem o desenvolvimento do projeto em questão de acordo com a arquitetura. Também serão descritos os principais trabalhos relacionados.

2.1. Arquitetura Geral

A Figura 2.1 representa a arquitetura geral do framework:

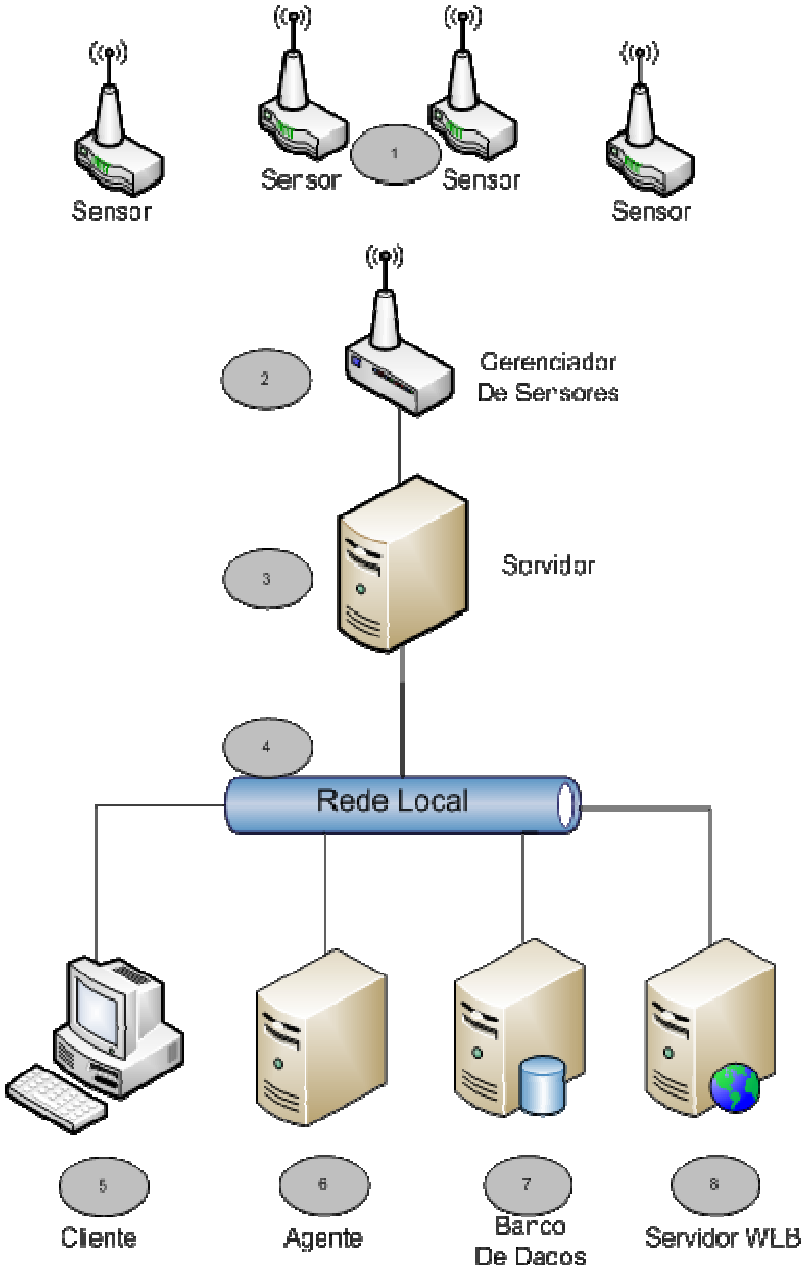


Figura 2.1 Arquitetura Geral do *Framework*

Os Sensores representados por (1) têm a função de estar sempre monitorando os fenômenos a serem analisados, no ambiente em que estão inseridos. Todos eles devem repassar as informações para o módulo Gerenciador de Sensores (2), que imediatamente as repassa para o Servidor (3). Esse Servidor é responsável por se comunicar com (2), tratar as informações advindas do mesmo e as disponibilizar em Rede Local (4).

Os Clientes (5) são aplicativos que requisitam as informações disponibilizadas em (4), por (3). Já o Agente (6) tem a função de inserir essas informações no Banco de Dados (7), que podem servir de histórico para uma análise mais aprofundada das oscilações observadas no ambiente em que o sensor esteja presente.

O Servidor WEB (8) é responsável por manter uma página, que adquire as informações sensoriais na Rede Local, e as disponibiliza na internet. Também tem a função de oferecer alguns serviços: cadastro e remoção de usuários, com suas devidas autorizações; cadastro e remoção de alarmes, caso informações sensoriais não estejam de acordo com o esperado; e a apresentação dos gráficos referentes ao histórico de cada sensor a partir das informações armazenadas em (7).

O Banco de Dados tem a função de armazenar as informações advindas tanto do Agente (6), quanto da página mantida pelo Servidor WEB (8).

Com isso, a arquitetura pode ser dividida em três áreas principais:

- Redes sem fio: Responsável pela comunicação entre os Sensores (1) e o Gerenciador de Sensores (2).

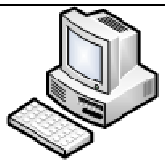
- Sistemas de Coordenação: Tem a função de facilitar a interação entre dispositivos dentro da Rede Local (4), controlando a entrada e saída de equipamentos. Os recursos são disponibilizados pelo Servidor (3), e consumidos por Clientes (5), pelo Agente (6), e pelo Servidor WEB (8).

- Desenvolvimento WEB: O Servidor WEB (8) manterá uma página, que terá como função adquirir os dados junto a Rede Local (4) e oferecer alguns serviços para todo o mundo, através da internet.

2.1.1. Descrição dos Módulos

A Tabela 2.1 descreve as responsabilidades de cada módulo dentro da arquitetura geral do framework.

Tabela 2.1 Descrição dos módulos da arquitetura geral do *framework*

 Sensor	Módulo responsável por repassar as informações sensoriais para o Gerenciador de Sensores.
 Gerenciador de Sensores	O Gerenciador de Sensores é o responsável pelo recolhimento dos valores de todos os Sensores, e por enviá-los para o Servidor.
 Servidor	Responsável pela aquisição junto ao Gerenciador de Sensores das informações advindas de cada Sensor. Também tem como função disponibilizar em rede local essas informações.
 Agente	Consiste em uma aplicação que requisita a todo o momento as informações sensoriais e as armazena no banco de dados.
 Servidor WEB	O Servidor WEB é o hardware responsável pelo container IIS. Este, por sua vez, é o mantenedor da aplicação WEB.
 Clientes	Software específico que funciona apenas dentro da rede local, pois requisita diretamente ao Servidor as informações sensoriais.
 Banco de Dados	O banco de dados é responsável pelo armazenamento das informações relevantes.

2.2. Tecnologias Relacionadas

Aqui serão apresentadas as tecnologias relacionadas que podem vim a viabilizar a implementação da arquitetura, descrita em 2.1. Existem três áreas macro no sistema que será desenvolvido, e para cada uma delas serão detalhadas as características a respeito das tecnologias relacionadas, mostrando as vantagens e desvantagens. A partir dessa descrição, poder-se-á decidir quais serão utilizadas de acordo com o perfil do projeto.

2.2.1. Redes sem Fio

Os padrões de comunicação sem fio surgiram para fornecer maior flexibilidade, em substituição aos sistemas cabeados. Para a transferência de dados entre os sensores serão estudados nesse trabalho alguns desses padrões mais importantes, como Wi-Fi, *Bluetooth* e *ZigBee*.

2.2.1.1. Wi-Fi

Wi-Fi, abreviatura de "*wireless fidelity*" e também conhecida como *Wireless* LAN (WLAN), é marca registrada pertencente à *Wireless Ethernet Compatibility Alliance* (WECA). Trata-se de uma rede local sem fio padronizada pelo IEEE 802.11.

Redes WLAN têm como principais aplicações:

- Redes locais internas de residências, empresas, lojas e escritórios, complementando ou até mesmo substituindo redes que utilizam cabeamento, seja por praticidade ou necessidade, visto que a instalação de cabos e fios é dificultosa em certos locais.

- Redes públicas de acesso à internet são mais conhecidas pelo nome de Wi-Fi.

2.2.1.1.1. Padrões

A padronização das redes LAN e MAN, locais e metropolitanas respectivamente, é liderada pelo Comitê 802 do IEEE.

Atualmente existem algumas alternativas aperfeiçoadas do padrão 802.11 inicial, criado em 1999. A Tabela 2.2 ilustra as principais características dos padrões *Wi-Fi* mais utilizados.

Tabela 2.2 Principais características dos padrões Wi-Fi mais utilizados

	802.11b	802.11g	802.11n	802.11a
Freqüência	2400-2483,5 MHz			5150-5350 MHz 5470-5725 MHz* 5725-5850 MHz
Técnica de Modulação	DSSS	DSSS, OFDM	DSSS, OFDM	OFDM
Taxa de Dados	Máximo de 11Mbps**	Máximo de 54Mbps	Superiores a 350Mbps	Máximo de 54Mbps

* O IEEE 802.11h estende este padrão.

**Existe um aperfeiçoamento a esta norma que permite alcançar taxas de até 44 Mbps.

O padrão 802.11n ainda está sendo definido pela IEEE podendo atingir velocidades superiores a 350Mbps, o terceiro *draft*[4] deste novo padrão foi lançado em novembro de 2007, espera-se que poucas mudanças sejam adicionadas em sua primeira versão oficial. Desta forma muitos fabricantes já lançaram equipamentos com implementações próprias dos 802.11n comprometendo-se a lançar *firmwares*[5] de atualização para seus equipamentos quando o padrão final for lançado.

Embora tenham um custo de produção um pouco maior se comparados aos 802.11a/b/g, os produtos 802.11n tendem a cair rapidamente de preço e a substituir os padrões mais antigos, por oferecem vantagens incontestáveis em relação à velocidade de transmissão e ao alcance máximo. O ganho de velocidade tende a variar de acordo com o modelo do produto e com o fabricante, mas (com exceção de alguns equipamentos baseados no *draft* 1.0) existe um ganho expressivo em relação às redes 802.11a/b/g.

Vantagens

Possui como principais vantagens:

- **Taxa de Transferência**

  Possui taxas de transferência relativamente altas como mostrado na Tabela 2.1, principalmente se for levado em consideração o novo padrão 802.11n que supera com facilidade o padrão Ethernet de 10Mbps e até mesmo o Ethernet de 100Mbps (o mais popular atualmente para redes cabeadas).

- **Alcance**

  Possui um alcance médio. No padrão g é possível atingir em campo aberto algo em torno de 200 metros, enquanto em campo fechado se alcança no máximo um raio de 50 metros. Antenas podem ser agrupadas ao sistema para aumentar essa distância, porém além de se tornar mais custoso ainda elevaria bastante o consumo de energia.

- **Segurança**

  Através do antigo protocolo *WEP*[6], e o mais recente chamado WPA[7], que surgiu para suprir as limitações de WEP. A segurança em redes Wi-Fi se encontra em um nível satisfatório, porém devido a seu alto uso em situações críticas, como transações financeiras por exemplo, muito ainda precisa ser pesquisado.

Desvantagens

- **Custo**

  Apesar de não haver custos para a utilização do padrão Wi-Fi, os custos de aquisição dessa tecnologia ainda é um pouco alta, devido a sua taxa de transferência ser relativamente grande.

---

[4] Draft – Documento contendo especificações preliminares de um determinado padrão.
[5] Firmware – Software embarcado responsável pelo controle de um determinado dispositivo de eletrônico.
[6] *WEP - Wired Equivalent Privacy*, faz parte do padrão IEEE 802.11 (ratificado em Setembro de 1999). Consiste em um protocolo para proteção de redes sem fios.
[7] WPA - Wi-Fi *Protected Access*. Protocolo de segurança para redes sem fio desenvolvido para substituir o protocolo WEP, devido a suas falhas de segurança.

- **Consumo**

  Apresenta um consumo de energia bastante relevante devido a sua alta banda. Isso torna inviável a alimentação por baterias.

2.2.1.2. Bluetooth

Bluetooth é uma tecnologia de comunicação sem fio de baixo custo que começou a ser desenvolvida em 1994 pela Ericsson, e a partir de 1998 pelo *Bluetooth Special Interest Group* (SIG), consórcio estabelecido inicialmente pela Ericsson, IBM, Sony, Toshiba, Intel e Nokia. Atualmente esse consórcio inclui mais de 2000 empresas e tem sua maior utilização na comunicação entre pequenos dispositivos, telefones celulares, Pocket PCs, PDAs. Também é utilizado para a comunicação de periféricos, como scanners, impressoras, e qualquer dispositivo que possua um *chip bluetooth*. A tecnologia opera dentro da faixa aberta de 2,4 GigaHertz com alcance variável dependendo da categoria e da especificação. A comunicação entre dispositivos *bluetooth* de classe 1 pode atingir até 100 metros. Enquanto que a transmissão entre dispositivos de classe 2 dificilmente ultrapassa 10 metros.

Segundo [14] cada dispositivo *bluetooth* é dotado de um número único de 48 bits utilizado na identificação. São possíveis conexões de até 8 dispositivos, desde que um deles seja mestre (dispositivo principal) e os outros escravos. A faixa de freqüência do *bluetooth* é dividida em 79 portadoras espaçadas de 1 MegaHertz, dessa forma cada dispositivo pode transmitir em 79 diferentes frequências, minimizando as interferências.

O dispositivo principal depois de sincronizado com os demais pode mudar as freqüências de transmissão dos dispositivos escravos, até 1600 vezes por segundo (isso é conhecido como *frequency hopping* ou saltos de freqüência). A tecnologia possui uma banda teórica de 2Mbps e efetiva de 721 Kbps.

Vantagens

- **Segurança**

  A tecnologia *bluetooth* oferece transmissão criptografada de seus dados utilizando o protocolo *WEP*.

- **Consumo**

  Apresenta um consumo relativamente baixo se comparado ao Wi-Fi, principalmente quando se encontra no modo *standby*. Porém, no momento em que está enviando ou recebendo dados, o consumo chega a se equiparar.

- **Custo de aquisição**

  Com a difusão cada vez maior de dispositivos *bluetooth* no mercado, o custo dessas interfaces torna-se cada vez menor.

- **Custo de utilização**

  Por ser uma tecnologia ponto a ponto de curto alcance, onde os dados só trafegam entre o dispositivo que iniciou a conexão e o outro dispositivo onde ele está conectado, a utilização do *bluetooth* é isenta de custos.

Desvantagens

- **Alcance**

  O alcance dos dispositivos *bluetooth* é limitado a uma distância de 100 metros no caso de dispositivos de classe 1, entretanto devido ao alto consumo de energia dos dispositivos que atendem a esse padrão, a maioria dos dispositivos móveis utiliza interfaces *bluetooth* de classe 2, limitando o alcance a apenas 10 metros.

- **Taxa de transferência**

  Possui uma taxa de transferência razoável, aproximadamente 1Mbps. Essa banda pode permitir até mesmo streaming de vídeo (de baixa qualidade).

- **Limite de usuários**

  A especificação do protocolo *bluetooth* permite apenas 7 usuários conectados a 1 dispositivo principal.

2.2.1.3. *ZigBee*

No final de 2004, um conjunto de empresas de diferentes segmentos do mercado chamada "*ZigBee Alliance*" [5] definiu o padrão *ZigBee*. Mais de 200 empresas fazem parte hoje da aliança como Philips, Motorola, Siemens, Bosch e etc. Trata-se de um protocolo para redes de sensores, projetado para permitir comunicação sem fio confiável e com baixo consumo de energia.

Baseada no padrão IEEE 802.15.4, o *ZigBee* opera em três bandas de rádio conhecidas como ISM (*Industrial, Scientifical and Medical*), as quais estão isentas de licenciamento. A taxa de transmissão possível varia com a banda: nos Estados Unidos, com uma banda de 915Mhz podem ser obtidos até 40Kbps, com 10 canais de comunicação; na Europa tem-se uma banda de 868Mhz, com uma taxa de 20Kbps, e apenas 1 canal; já em todo o resto do mundo, a banda é de 2.4Ghz, que pode atingir até 250Kbps, com 16 canais.

Vantagens

- **Ciclo de Trabalho**

  O *ZigBee* apresenta um ciclo de trabalho muito baixo, ou seja, a fração de tempo em que ele está ativo é mínimo. Isso possibilita maior autonomia quando alimentado por baterias.

- **Estados de Operação**

  Suporta dois estados de operação: inativo ("*Sleep Mode*") e ativo, sempre com fluidez na passagem de um para o outro.

- **Sincronização**

  Permite que os dispositivos permaneçam em "*Sleep Mode*" sem necessidade exigente de sincronização.

- **Comunicação**

  Capacidade de permanecer longos períodos sem comunicação.

- **Suporte a várias Topologias**

  Oferece suporte a topologias de rede tanto estáticas e dinâmicas, sejam elas em estrelas, em árvores ou em malhas.

- **Segurança**

  Recurso de encriptação de 128 bits.

- **Custo de Utilização**

  Como a comunicação ocorre entre módulos que suportam *ZigBee*, a utilização é gratuita.

- **Quantidade de Nós**

  Possibilidade de utilização de redes com até 65535 nós para cada nó coordenador, procurando garantir sempre baixa latência.

- **Alcance**

  O seu alcance depende diretamente da potência dos equipamentos que implementam o protocolo *ZigBee*, podendo chegar até a 1,6Km. Logo, quanto maior o alcance maior também a potência necessária, aumentando assim o consumo de energia.

Desvantagens

- **Taxa de Transferência**

  Possui uma baixa taxa de transferência, aproximadamente 200kbps.

- **Custo**

  Por ser uma tecnologia mais recente que as demais, o custo de aquisição de um dispositivo *ZigBee* ainda está relativamente alto.

- **Baixa Complexidade**

  A pilha protocolar de implementação do *ZigBee* é pequena em termos de complexidade, exigindo menores recursos nos dispositivos que a utilizem. Porém, isso conduz a interfaces de baixo custo.

2.2.1.4. Breve Comparação

A Tabela 2.3 mostra as características principais de cada uma das tecnologias de rede sem fio. Essa comparação será decisiva para a escolha da tecnologia a ser utilizada nessa monografia.

Tabela 2.3 Comparação das tecnologias de rede sem fio relacionadas

Especificação	Banda	Consumo	Pilha Protocolar	Vantagens	Principais Aplicações
Wi-Fi IEEE 802.11bg	54Mbps	> 400 mA TX, standby: 20mA	1Mb	Elevada taxa de transferência	Internet; Transferência de arquivos; Vídeo/Áudio
Bluetooth **IEEE 802.15.1**	1Mbps	> 400 mA TX, standby: 0.20mA	≈250Kb	Interoperabili-dade; Ausência de cabeamento	Periféricos de PC; Celulares; PDA's
ZigBee **IEEE 802.15.4**	200Kbps	> 30 mA TX, standby: 20µA	≈32Kb	Consumo; Latência; Número de Nós	Sensores; Dispositivos alimentados por bateria

Na Figura 2.2 é possível visualizar em termos gráficos a relação entre o alcance e a taxa de transmissão das tecnologias relacionadas.

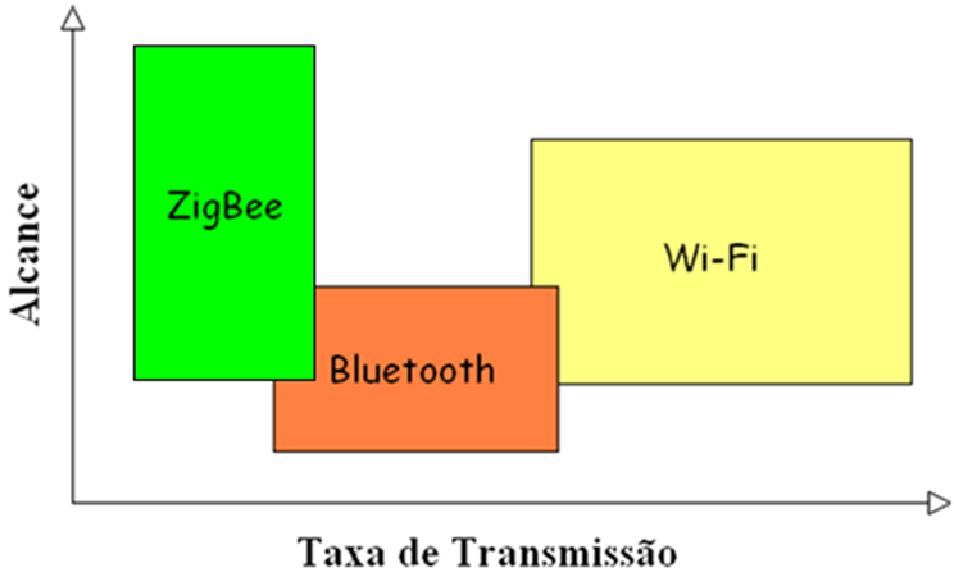


Figura 2.2 Representação gráfica da taxa de transmissão X alcance das tecnologias relacionadas

2.2.2. Sistemas de Coordenação

As redes são dinâmicas por natureza, mudam a todo instante, com adição e remoção de dispositivos, atualização de sistemas existentes e conserto de partes da rede em falha. Sistemas de coordenação de dispositivos têm que ser adaptáveis e flexíveis para oferecer interação entre os mesmos.

Dois dos principais sistemas de coordenação que estão emergindo atualmente são Jini e UPnP. Características que devem estar presentes nos mesmos são: anunciação de sua presença

na rede; auto-configuração; descoberta automática de dispositivos na rede; interoperabilidade; e habilidade de oferecer, reconhecer e consumir serviços na rede.

2.2.2.1. Jini

Jini, sigla para Interface de Rede em Java, foi criado em 1994/1995 pela *Sun Microsystems* para conectar universalmente vários aparelhos. Originada de trabalhos acadêmicos, essa tecnologia surgiu para tornar simples a computação distribuída, através da tecnologia *plug and play*. Ela permite criar uma rede dinâmica de comunicação entre serviços oferecidos, que pode ser memória, espaço em disco, um dispositivo, entre outros.

Esse sistema de coordenação disponibiliza e requisita serviços na rede, criando interações espontâneas entre as aplicações. Tais serviços se comunicam usando um protocolo, que em essência são interfaces Java. Jini utiliza o conceito de federação na rede, chamada de Djin, onde todas as entidades que integram formam um conjunto igualitário em termos de autoridade. Pode-se ainda nominar grupos de serviços que tenham certa afinidade, de modo a facilitar a busca e restringir o acesso.

Jini oferece um serviço chamado *Lookup* em tempo de execução, que armazena todos os recursos oferecidos no momento e serve para facilitar a interação entre dispositivos que acessem a rede. Ele coloca todos os serviços à disposição dos clientes, que querem consumi-los ou gerenciá-los.

A comunicação entre dispositivos é realizada através de RMI (*Remote Method Invocation*), que permite a um programa escrito na linguagem Java acessar métodos remotos de outros programas desenvolvidos em Java, através de uma interface conhecida. Ou seja, os dispositivos comunicantes precisam ter Java embutido para que tudo ocorra perfeitamente.

Existe a possibilidade de dispositivos que não utilizam Java também se conectarem à rede, através de *gateways*, *proxies* e *bridges* fazendo a conversão de protocolo. Porém, introduzir dispositivos dessa maneira não é muito recomendável, pois pode inserir falhas na rede.

Vantagens

- **Entrada na rede**

  No modo *Unicast* e *Multcast* para o serviços *lookup* de Jini. A inscrição é feita logo a posteriori, se o dispositivo for reconhecido.

- **Descoberta de dispositivos**

  A partir da inscrição na rede, questionar o *lookup* com as propriedades dos dispositivos que se deseja encontrar.

- **Descrição das capacidades**

  Duas fases são necessárias: padronização das informações que serão inscritas, e só então a padronização das interfaces RMI. Os dados que serão inscritos em geral têm propriedades de fácil verificação.

Desvantagens

- **Autoconfiguração**

  O Jini não oferece suporte a autoconfiguração, um dispositivo IP quando adicionado a rede terá que ser configurado com um endereço IP. Ou seja, o *lookup* só poderá registrar seus serviços mediante um IP já fixado.

- **Chamada de serviços**

  Java não oferece muita independência em relação aos *drivers*. Os métodos nas Interfaces RMI devem estar bem padronizados para que a comunicação funcione corretamente.

- **Segurança**

  Jini não trata de segurança, transfere a obrigação para Java e RMI.

2.2.2.2. UPnP

UPnP – Universal Plug and Play – é uma tecnologia baseada num conjunto de protocolos e criada em 1999 pela Microsoft para ser utilizada em casas inteligentes, escritórios e outros tipos de redes locais. Atualmente seu desenvolvimento e especificação são regidos pelo fórum *UPnP* [6], uma organização independente com mais de 770 membros.

Através do *UPnP*, dispositivos podem dinamicamente entrar em uma rede, conseguir um endereço IP, disseminar seus recursos/serviços pela rede e obter conhecimento da presença e dos recursos/serviços de outros dispositivos automaticamente. Permitindo assim a criação de redes de configuração zero. Após a apresentação e reconhecimento mútuo de serviços entre os dispositivos, pode ser criada uma conexão de rede ponto a ponto entre dispositivos.

O *UPnP* é baseado em *IP* e utiliza protocolos padrões como o *HTTP*[8], *XML*[9] e *SOAP*[10], o que lhe permite adequar-se de forma transparente as redes existentes e transformar a interoperabilidade em um recurso praticamente nativo de sua arquitetura. Seu escopo de atuação é tão diversificado que pode abranger cenários tão distintos quanto, automação doméstica, redes de automóveis, entretenimento digital e robótica.

A inspiração técnica que antecedeu o desenvolvimento do *UPnP* buscou a criação de um *framework* computacionalmente distribuído baseado em tecnologias Web e focado em pequenas redes, especialmente redes domésticas. Nesse contexto foram criados dois protocolos: o *SSDP* (*Simple Service Discovery Protocol*), protocolo de descoberta que reutiliza cabeçalhos HTTP; e o *GENA* (*General Event Notification Architecture*), "publique e assine" baseado no *HTTP*.

A arquitetura de dispositivos *UPnP* é o núcleo a partir do qual os dispositivos e a especificação de serviços são construídos. A arquitetura *UPnP* define dois tipos de *hosts*: "*Control Points*" (Pontos de Controle) que são os clientes e "*Devices*" (Dispositivos).

[8] HTTP (*HyperText Transfer Protocol*) – Protocolo utilizado para transferência de documentos na WEB.
[9] XML (*eXtensible Markup Language*) – Linguagem universal para permitir a troca de informações de forma estruturada atráves da Internet.
[10] SOAP (*Simple Object Access Protocol*) – Protocolo de transporte de mensagens independente, e cada mensagem *SOAP* é um documento *XML*.

Dispositivos podem incluir serviços e podem ser embarcados em outros dispositivos. A arquitetura do *UPnP* tem seis características principais:

- Endereçamento: protocolos de autoconfiguração IPv4 e IPv6;
- Descoberta: SSDP;
- Descrição: XML
- Controle: SOAP;
- Eventos: GENA;
- Apresentação: HTML e extensões de fabricantes.

Vantagens

- **Independente de Linguagem**

  O protocolo *UPnP* é independente de linguagem de programação podendo ser escrito em qualquer linguagem desde que siga as especificações do padrão.

- **Multiplataforma**

  O *UPnP* não está limitado a uma única plataforma ou sistema operacional. Pode ser executado em qualquer sistema operacional e principalmente em dispositivos embarcados.

- **Configuração zero**

  UPnP é baseado no mecanismo de configuração zero. De acordo com [9] esse tipo de tecnologia não necessita de uma configuração prévia para entrar em funcionamento, bastando ser conectado a uma rede local para executar suas atividades.

Desvantagens

- **Escalabilidade**

  O *UPnP* não especifica um serviço de registro de dispositivos e a comunicação para endereçamento e descoberta de novos dispositivos/serviços é baseada em mensagens enviadas por *broadcast*. Dessa forma se existirem muitos dispositivos *UPnP* em uma sub-rede, uma grande parte da largura de banda será ocupada apenas por mensagens em *broadcast* dos dispositivos.

- **Segurança**

  A especificação original do *UPnP* não inclui mecanismos de segurança de dados. Ele assume que a segurança necessária será provida por outros meios como, 802.11 *WEP*, WPA[11], NAT[12], *firewalls* e VPNs[13]. Essa falta de padrão de autenticação, autorização

[11] *WPA* - Wi-Fi *Protected Access*, surgiu em 2003 do esforço conjunto de membros da Wi-Fi Alliance e membros do IEEE, empenhados em aumentar o nível de segurança das redes sem fio, combatendo algumas vulnerabilidades do *WEP*.

[12] *NAT - Network Address Translation*, técnica que consiste em reescrever os endereços IP de origem de um pacote que passam sobre um roteador ou *firewall* de maneira que um computador a partir de uma rede interna tenha acesso ao exterior (uma rede pública como a Internet, por exemplo).

e comunicação segura de dados é talvez o maior problema da arquitetura *UPnP*. A ausência da especificação de requisitos de segurança fez com que a primeira geração de produtos compatíveis com o protocolo tivessem o *UPnP* desabilitado por usuários mais cautelosos. Em alguns casos os próprios fabricantes desabilitavam a funcionalidade *UPnP*, deixando a cargo dos usuários habilitá-la ou não.

2.2.2.3. Breve Comparação

Jini e UPnP são os sistemas de coordenação que mais vêm crescendo ultimamente. São tecnologias que têm conceitos semelhantes: provedores de uma plataforma flexível para API's e ferramentas. Porém, enquanto UPnP se utiliza de protocolos abertos já disponíveis no mercado, Jini desenvolveu o seu próprio protocolo.

Para um sistema de coordenação funcionar corretamente é preciso inserir padronização na operação dos mesmos. Porém, faz-se necessário achar o ponto de equilíbrio entre padronização e a autonomia de dispositivos, uma vez que são dois atributos interligados. Enquanto Jini valoriza mais a padronização (criação de seu próprio protocolo), UPnP tem como principal característica sua autonomia (utilização de protocolos abertos).

A principal diferença é como Jini e UPnP organizam suas API's. Em Jini, os serviços precisam usar uma API standard generalizado para ser útil a um cliente que conhece esta API, em Java. Em UPnP, os serviços podem fazer total uso da plataforma em quês estão baseadas, de modo que as plataformas individuais mapearão a API para a API nativa. Uma evolução do UPnP é a característica de autoconfiguração.

2.2.3. Desenvolvimento WEB

Para difundir as informações advindas dos sensores para fora da rede local, por todo o mundo através da internet, é preciso fazer a escolha de uma plataforma para desenvolvimento WEB. Características como integração com o sistema, velocidade e segurança serão aqui analisados, apresentando as vantagens e desvantagens. Logo após ocorrerá uma breve comparação entre as tecnologias mais utilizadas dentro desse âmbito: PHP, JSP e ASP.NET.

2.2.3.1. PHP

PHP, um acrônimo recursivo para *Hypertext Preprocessor*, teve sua primeira versão lançada em 1995 por Rasmus Lerdorf. Trata-se de uma linguagem de programação orientada a objetos para criação de páginas WEB. De fácil aprendizado e simples utilização, essa tecnologia originalmente foi criada para o desenvolvimento de pequenos scripts dinâmicos. Atualmente se encontra na versão 5, porém com a versão 6 já em desenvolvimento, sempre com recursos adicionados. Essa facilidade na evolução de PHP dá-se por ela ser *Open Source*, ou seja, uma plataforma de código aberto onde todos podem acessar e contribuir.

[13] *VPN* - Virtual Private Network, rede de comunicações privada normalmente utilizada por empresas ou instituições, construída em cima de uma rede de comunicações pública como, por exemplo, a Internet.

Vantagens

- **Velocidade**

  O PHP é bastante rápido, além de ser robusto.

- **Memória**

  Gasta-se pouca memória com PHP.

- **Suporte a imagens e documentos**

  Oferece suporte para imagens GIF e documentos PDF.

- **Banco de dados**

  Pode-se acessar várias base de dados, como *Oracle*, MSSQL, *Firebird* e MySQL.

- **Orientada a objetos**

  Oferece suporte a orientação de objetos, com ganho em produtividade.

- **Portabilidade**

  Executa em qualquer em qualquer plataforma.

- **Integração com HTML**

  É perfeitamente possível fazer a fusão de PHP com HTML.

- **Custo**

  Como PHP é código aberto, é possível utilizá-lo sem nenhum custo. Além disso, está em constante atualização, com falhas sendo corrigidas.

Desvantagens

- **Tipagem**

  PHP oferece poucos tipos, possuindo uma tipagem fraca.

- **Extensibilidade**

  PHP tem pouca extensibilidade, com um conjunto pré definido de comandos, sendo possível definir novos em alguns poucos casos.

- **Documentação**

  A documentação de PHP é incompleta.

2.2.3.2. JSP

JavaServer Pages (JSP) é uma linguagem para desenvolvimento WEB que se baseia em Java, lançada em 1999 pela *Sun Microsystems*. Segundo [16], essa tecnologia permite ao desenvolvedor de páginas a criação de aplicações que acessem banco de dados, a manipulação de arquivos no formato de texto, além de extrair dados a partir de formulários e captar informações sobre o usuário e sobre o servidor.

Vantagens

- **Portabilidade da plataforma**

  Por ser baseada em Java, pode ser executada em diversos sistemas operacionais, como Windows e Linux.

- **Linguagem de alto nível**

  Possui uma sintaxe muito parecida com a de Java.

- **Separação entre o código e a interface gráfica**

  Separação entre a parte estática (programação visual) e dinâmica (programação lógica). Ou seja, permite o desenvolvimento em paralelo da equipe responsável pela definição da interface gráfica e da equipe imbuída de implementar a lógica do negócio.

- **Segurança**

  Considerada bastante segura, utilizada até em páginas de bancos e corporações.

Desvantagens

- **Curva de aprendizado**

  Apesar de JSP ser uma linguagem de alto nível, se o programador ainda não tiver experiência com Java, a curva de aprendizado pode causar atrasos no projeto.

- **Tratamento de erros**

  Encontrar erros com a sintaxe de JSP é uma tarefa relativamente complicada. Debugar também não é simples: o compilador traduz um arquivo JSP para .Java, e mediante a ocorrência de um erro não mostra-se a linha do arquivo .JSP, e sim do .Java, dificultando a resolução do problema. Porém, com o uso de uma IDE esse problema pode ser resolvido.

- **Integração com banco de dados**

  Apesar de suportar um grande número de banco de dados através de conexão JDBC, achar um *driver* e fazê-lo funcionar pode demandar bastante tempo do programador.

- **Documentação**

  Pouca documentação em português.

- **Performance**

  Páginas JSP são mais lentas que as principais concorrentes.

2.2.3.3. ASP.NET

ASP.NET, sucessor do ASP, é uma plataforma da Microsoft para desenvolvimento WEB. Essa tecnologia não é considerada por muitos uma linguagem de programação, e sim uma plataforma baseada no *Framework* .NET que permite a criação de páginas dinâmicas. Embora aplicações ASP.NET possam ser escritas em *notepad* com o uso do compilador .NET, essa tecnologia é geralmente utilizada em conjunto com o ambiente Visual Studio .NET.

Vantagens

- **Extensibilidade**

  O ASP.NET foi originalmente lançado como uma estrutura aberta. Ou seja, muitos módulos podem ser construídos, modificados ou até substituídos.

- **Orientada a objetos**

  É possível utilizar todas as linguagens da plataforma .NET, por exemplo C# e *Visual Basic* .NET. Essa característica representa um enorme ganho de produtividade, acrescentando inúmeras vantagens como a possibilidade de se criar controles, definir tipos para objetos e variáveis, além de criar procedimentos e funções.

- **Separação entre o código e a interface gráfica**

  Em ASP.NET os arquivos de código e de telas ficam separados. Isso permite que equipes distintas trabalhem separadamente, também aumentando o ganho de produtividade.

- **Reconhecimento de dispositivos**

  ASP.NET pode reconhecer uma variedade de dispositivos e processar a marcação apropriada para eles.

- **Tratamento de erros**

  Por se utilizar de uma linguagem de alto nível da plataforma .NET, essa tecnologia incorpora todos os benefícios, como depuração e fácil tratamento de erros.

- **Controles de servidor**

  Controles para a utilização nas páginas ASP.NET, também conhecidas como *Webforms*, possibilitando o uso de eventos.

- **Desempenho**

  Os códigos são compilados no servidor de forma a gerar uma página HTML no navegador de quem está acessando, aumentando o desempenho.

Desvantagens

- **Flexibilidade**

  A utilização desta tecnologia é gratuita, porém a plataforma .NET não é aberta.

- **Banco de dados**

  Não tem a integração com muitas bases de dados, porém existe a possibilidade da conexão ODBC.

2.2.3.4. Breve Comparação

As três tecnologias mencionadas para desenvolvimento WEB apresentam características positivas e negativas. A Tabela 2.4 mostra os principais fatores que podem influenciar na decisão da linguagem a ser utilizada.

Tabela 2.4 Principais características das tecnologias relacionadas para Desenvolvimento WEB

Linguagem	Características Positivas	Características Negativas
PHP	*Open source*; Velocidade e memória	Fraca tipagem; Pouca extensibilidade
JSP	Segurança; Linguagem de alto nível	Performance; Integração com banco de dados
ASP.NET	Orientada a objetos; Reconhecimento de dispositivos	Flexibilidade

2.3. Trabalhos Relacionados

A área de pesquisas envolvendo redes de sensores sem fio é bastante ativa e vem crescendo em todo o mundo, incentivada por grandes investimentos. Devido à grande diversidade de artigos relacionados às RSSFs, serão apresentados apenas os mais relevantes. Também serão apresentados trabalhos envolvendo outras tecnologias abordadas nesta monografia, como sistemas de coordenação de dispositivos em rede local.

Para a comunicação dos sensores com transmissão de dados sem fio, pode-se encontrar a descrição formal da tecnologia Wi-Fi em [17], da *Bluetooth* em [15] e do padrão *ZigBee* em [4]. Em [19], é traçado um comparativo entre essas e outras tecnologias *wireless*, apresentando as principais diferenças em relação às aplicações, ao alcance, à freqüência de operação, à máxima potência transmitida, à banda passante e aos tipos de dados de envio/recebimento, à diversidade de topologias que cada uma funciona e ao custo de aquisição destas tecnologias.

Neste documento também é apresentado uma visão geral de cada tecnologia, levando-se em conta a duração da bateria, a complexidade e a quantidade de dispositivos em uma mesma rede. No estudo, chegou-se a conclusão que para a comunicação de sensores a melhor opção é o padrão *ZigBee*, por ter características de baixo custo, pequeno consumo de energia, alta eficiência utilizando poucos recursos de processamento, segurança de dados e o determinismo na rede. Porém, o principal fator que levou a utilização desta tecnologia é a pequena taxa de dados transmitida, pela banda passante não ser crítica no projeto. Tal projeto consiste na comunicação de sensores presentes em uma planta industrial onde basicamente trafegam dados.

Em [42] realizou-se um estudo com a utilização do padrão *ZigBee* em uma rede de sensores sem fio na área industrial de celulose. Mostrou-se como se pode adequar corretamente RSSFs neste tipo de indústria. Em [43] foram mencionadas algumas limitações na utilização deste padrão, discutindo os prós e contras para a transmissão entre dois tipos de imagens (JPEG vs. JPEG-2000), que se diferem por exemplo no modo de compressão. Um módulo *ZigBee* foi plugado a um PC e age como um coordenador, semelhante ao desenvolvido neste trabalho de graduação. Ele é responsável pela aquisição de informações como rotas e topologias da rede. O software executado no PC é bastante simples, apenas requisitando para si a transferência de imagens de um dispositivo *ZigBee*. Mostrou-se que a formatação JPEG-2000 é mais resistente a erros, tornando-se mais viável em redes com baixa banda passante.

Já no artigo [31] descreve-se um sistema utilizando rede de sensores sem fio para o monitoramento de ambientes em salas limpas. O lugar para medição foi a sala de processos do Laboratório de Sistemas Integráveis da Universidade de São Paulo. O parâmetro ambiental escolhido foi a temperatura, como no caso desse trabalho de graduação. Ainda de acordo com [31], é possível garantir a reprodutibilidade e a qualidade dos dispositivos neste ambiente através desse monitoramento. O padrão para a comunicação sem fio foi o IEEE 802.15.4, também conhecido popularmente como *ZigBee*. A rede apresentou um funcionamento dentro do esperado com base na transmissão dos dados, utilizando-se de roteadores entre os nós.

Realizou-se uma pesquisa em [44] através de simulações, onde diversas configurações foram testadas para o padrão *ZigBee*, no intuito de atingir melhor desempenho nas áreas de consumo de energia e confiabilidade. Ao final da pesquisa, foram sugeridas algumas configurações que melhoraram o desempenho, bem como algumas mudanças na especificação do padrão. Já em [45], analisou-se o consumo de energia e performance do padrão *ZigBee* em função de seu ciclo de trabalho e sincronização.

Foram estudadas em [46] interferências causadas entre os padrões *ZigBee* e Wi-Fi, quando operando corretamente na faixa de 2.4GHz. Mostrou-se que o padrão IEEE 802.15.4 acaba sendo o maior prejudicado com a interferência, necessitando de pelo menos 7MHz de diferença entre as freqüências de operação para funcionar satisfatoriamente.

Em [26] foi construído um sistema de localização robusta e prática sobre redes sem-fio 802.11 (ou Wi-Fi) em larga-escala. O artigo relata que o uso desta tecnologia deveu-se a seu baixo custo e a sua ampla utilização em organizações comerciais. Também influiu o fato dos dispositivos móveis hoje em dia já virem embutidos com esse padrão de tecnologia sem-fio.

Fez-se uma avaliação do uso do padrão IEEE 802.15.1 (ou *Bluetooth*) nas redes de sensores sem-fio, em [25]. Chegou-se a conclusão através de experimentos que *Bluetooth* pode ser útil na comunicação entre sensores em um nicho de aplicações que troquem volumes indefinidos de dados durante um intervalo de tempo ilimitado. Porém, o autor do artigo acredita que é difícil prever o uso de *Bluetooth* como uma alternativa para a comunicação de nós em redes de sensores sem-fio. Em [47], o autor do artigo afirma que *Bluetooth* perdeu mercado para o padrão *ZigBee*.

Para o a disponibilização das informações sensoriais em rede local, como a temperatura de cada módulo *ZigBee* espalhado no ambiente, pode-se encontrar a descrição formal do conjunto de protocolos UPnP em [6]. Já outro concorrente, o Jini que também emerge no mercado, tem sua definição em [13].

Em [41] é proposto um framework onde clientes Jini podem consumir serviços oferecidos por dispositivos UPnP, bem como clientes UPnP podem requisitar serviços disponibilizados por dispositivos Jini. Essa interoperabilidade ocorre sem modificação no serviço ou na implementação dos clientes.

O trabalho realizado em [36] discute como sensores sem-fio de baixo custo podem ser embutidos em ambientes *Universal Plug and Play*. A base do sistema Sindrion, como é chamado, consiste em configurar uma rede *wireless* entre dispositivos periféricos e terminais de computadores dedicados. O objetivo desta conexão é transferir todo o processamento de dados do periférico para os terminais dedicados. Sindrion suporta comunicação sem fio e autonomia entre os nós pertencentes à rede. Através desse sistema, encontrou-se um modo de integrar dispositivos em um ambiente UPnP sem que os mesmos sejam previamente compatíveis com esta tecnologia. Os periféricos oferecem o serviço de processamento, e os terminais que funcionam como pontos de controle requisitam este serviço e realizam todo o processamento com os dados. Essa solução é bastante interessante, uma vez que se pode economizar processamento no sensor, minimizando assim seu consumo de energia.

Em [39], foi criada uma plataforma para integração de dispositivos já existentes em uma unidade de gerenciamento Jini. Esta arquitetura pode ser aplicada a vários tipos de cenários, dispositivos e comunicação entre protocolos. Com isso, dispositivos legados podem se integrar dentro do sistema de coordenação Jini.

Já em [40], foi desenvolvido o projeto RoSES - *Robust Self-Configuring Embedded Systems* - que tem como objetivo criar sistemas embarcados distribuídos inerentemente robustos, flexíveis e manuteníveis, suportando degradação suave via reconfiguração intra-serviço de software. Para alcançar o objetivo, uma infra-estrutura apropriada foi montada para facilitar a funcionalidade *plug and play* requerida para que os nós formem uma rede distribuída dinânima e *ad-hoc*. Investigou-se muitas tecnologias de *middleware* para suprir tal infra-estrutura de execução no controle da rede local, e a que mais se adequou foi a tecnologia Jini.

Em relação ao desenvolvimento WEB, que tem a responsabilidade de adquirir junto à rede local as informações de temperatura através de algum sistema de coordenação, não foi encontrado nenhum sistema de gerenciamento remoto similar.

3. Sistema de Disponibilização de Informações Sensoriais Térmicas via UPnP

Este capítulo tem como objetivos apresentar as tecnologias adotadas, as ferramentas utilizadas, além de descrever toda a arquitetura do sistema tanto para o hardware quanto para o software. Como caso de estudo, utilizou-se de sensores de temperatura, porém poderia qualquer outro tipo de sensor, variando apenas de acordo com a aplicação desejada.

3.1. Tecnologias adotadas

Aqui serão apresentadas as tecnologias escolhidas para desenvolver o sistema, e as motivações pelas quais foram adotadas.

3.1.1. Redes sem fio

Para redes sem fio foi escolhido o protocolo *ZigBee*, por ser uma tecnologia voltada especialmente para sensores, minimizando bastante o consumo de energia dos dispositivos. Duas características também foram essências para a tomada de decisão: o suporte a mais de 65000 nós dentro de uma mesma rede, com um mesmo coordenador; e o fato da largura de banda no projeto não ser essencial, já que a taxa de dados transmitida é mínima.

3.1.1.1. Hardware Selecionado

A empresa Maxstream fabrica módulos *transceiver* que suportam o protocolo *ZigBee* definido pela "*ZigBee Alliance*". Duas versões foram lançadas pela empresa, chamadas de XBee e XBee Pro. Ambas transmitem a informação por rádio freqüência, operando na banda de 2.4Ghz, com taxa de até 250Kbps.

Os dispositivos devem receber uma tensão de alimentação compreendida entre 2.8V e 3.4V para funcionar corretamente. Consomem pouquíssima corrente quando no modo *Sleep* (menor que 10µA), gastando quase nada de energia. Porém, no modo ativo (transmissão/recepção), esse consumo cresce um pouco mais, principalmente no XBee Pro. Os dois modelos são compatíveis em tamanho e em *firmware*, além de possuir a mesma pinagem. Também têm a mesma de quantidade de canais conversores analógico/digital, que serão responsáveis pela leitura dos sensores térmicos.

A seguir são apresentadas as principais diferenças entre os dois módulos:

- **XBee**

  Potência de saída: 1 mW

  Alcance em ambientes internos: 30m;

  Alcance em ambientes externos: 100m;

  Corrente de transmissão: 45 mA quando operado em 3.3 V;

Corrente de recepção: 50 mA quando operado em 3.3 V;

Corrente de transição entre inativo/ativo: menor que 10 µA;

Custo: Em torno de $19 USD.

- **XBee Pro**

Potência de saída: 60 mW

Alcance em ambientes internos: 100m;

Alcance em ambientes externos: 1,6Km;

Corrente de transmissão: 215 mA quando operado em 3.3 V;

Corrente de recepção: 55 mA quando operado em 3.3 V;

Corrente de transição entre inativo/ativo: menor que 10 µA;

Custo: Em torno de $32 USD.

Para esse trabalho de graduação foi escolhido o dispositivo XBee Pro, que consome um pouco mais de energia, mas oferece um alcance bem maior. Porém, como os dois modelos são compatíveis, a troca de um pelo outro não demandaria nenhum esforço adicional.

3.1.2. Sistemas de Coordenação

Já para a comunicação de dispositivos em rede local foi escolhido o conjunto de protocolos *Universal Plug and Play*. Com essa tecnologia, os próprios dispositivos podem construir suas APIs[14] independentemente de sistema operacional ou de linguagem de programação. UPnP oferece simplicidade e interoperabilidade, além de permitir a identificação automática e o controle remoto de equipamentos.

Apesar de ser baseado em padrões, o UPnP atende perfeitamente às necessidades dos dispositivos presentes na rede, sendo bastante flexível.

3.1.3. Desenvolvimento WEB

ASP.NET foi escolhido para o desenvolvimento da aplicação WEB. São muitas as vantagens: ganho de desempenho, com a utilização de orientação a objetos; separação entre o código e o visual, possibilitando que o desenvolvedor da camada de negócios e o programador da interface gráfica trabalhem paralelamente; fácil depuração, com a utilização de uma linguagem de alto nível na plataforma .NET; código compilado no servidor, gerando uma página HTML que é exibida no navegador do cliente, aumentando assim a performance das aplicações; e outra característica determinante para a escolha dessa linguagem é o poder de reconhecimento de dispositivos, onde a resolução da página se adequa à tela de quem está acessando a página.

[14] API - *Application Program Interfaces.*

3.2. Ferramentas utilizadas

Nesta seção serão descritas todas as ferramentas utilizadas para o desenvolvimento do Sistema de Disponibilização de Informações sensoriais via UPnP.

3.2.1. Microsoft Visual Studio .NET 2005

O Visual Studio .NET 2005 [10] foi desenvolvido para atender as necessidades de desenvolvimento de software. IDE de fácil utilização, oferece simples integração entre C# e ASP.NET, que serão utilizados nesse projeto.

3.2.2. Microsoft SQL Server 2005

Sistema de gerenciamento de banco de dados desenvolvido pela Microsoft possuindo ótimo desempenho, escalabilidade e robustez. É utilizado em projetos de todos os portes. No contexto desse trabalho é utilizado para armazenar sua base de dados, tendo como responsabilidade a manutenção e consistência de informações essenciais sobre usuários, alarmes, sensores e temperaturas.

3.2.3. IIS – Internet Information Service

IIS [12] é um servidor de páginas web criado pela Microsoft para seus sistemas operacionais. Depois do lançamento da plataforma *.NET* em 2002 o *IIS* ganhou também a função de gerenciar as aplicações desenvolvidas em *ASP.NET* [49], com estas mesmas características foi utilizado no sistema aqui desenvolvido.

3.2.4. X-CTU

A programação dos módulos XBees Pro foi realizada via X-CTU, software disponibilizado pela MaxStream gratuitamente. O programa oferece uma interface gráfica bem amigável conforme a Figura 3.1, o que facilitou bastante a configuração dos dispositivos.

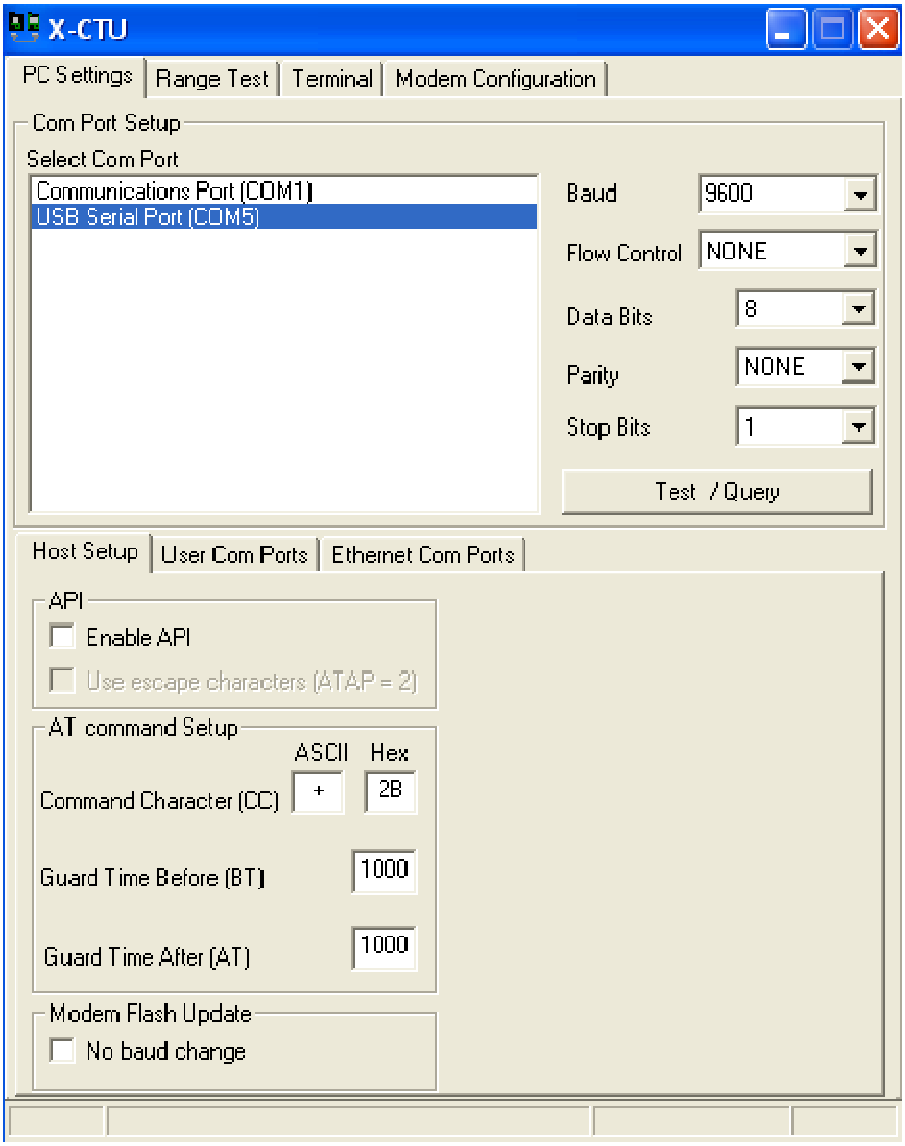


Figura 3.1 Print Screen do software X-CTU

A fim de facilitar a programação, uma placa conversora USB/Serial foi adquirida, chamada de CON-USBBEE (ver Figura 3.2). Em formato parecido com um *pen-drive*, basta plugá-la ao computador que o Windows cria uma conexão chamada de COMx, reconhecendo a placa como se fosse uma comunicação serial padrão RS232. Antes, é necessária a instalação de um *driver* USB a partir de um CD, que acompanhou a CON-USBBEE.

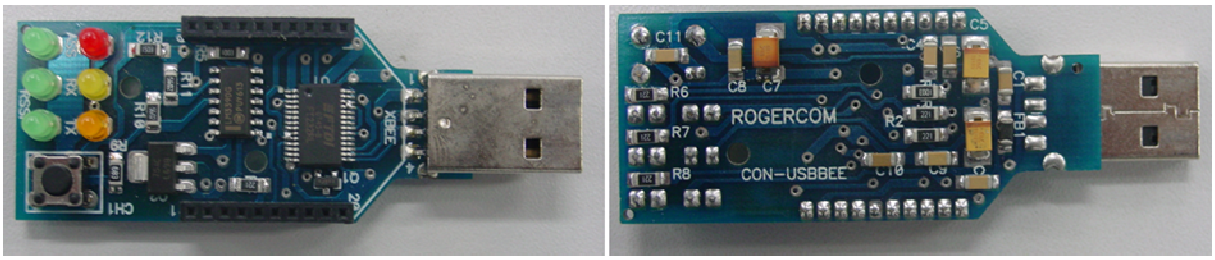


Figura 3.2 Visão Superior da placa CON-USBBEE (à esquerda); Visão Inferior da placa COM-USBBEE (à direita)

Essa placa pode configurar tanto dispositivos XBee como XBee Pro, pois os dois são compatíveis em tamanho e em software. Basta apenas conectá-la ao PC e iniciar o software X-CTU na serial (COMx) correspondente.

3.2.5. Eagle

Para a utilização do XBee Pro no projeto, foi necessária a construção de uma placa que o alimentasse e o protegesse de picos de tensão (tensões acima de 3.3V não são suportadas pelo dispositivo). Primeiro se constrói o esquemático, onde o objetivo é definir o circuito e conseqüentemente suas ligações. Depois se confecciona o *layout*, que define o design final da placa. A partir desse design é que se prototipa a placa propriamente dita.

O software Eagle (Figura 3.3 e Figura 3.4) desenvolvido pela Cadsoft, versão 4.16, foi utilizado tanto para a confecção do esquemático quanto para o *layout*. A escolha deveu-se à facilidade de uso, minimizando o tempo de projeto. O software também oferece flexibilidade, pois dispositivos eletrônicos que não estão na sua biblioteca, podem ser construídos de maneira prática e fácil. Abaixo segue algumas fotos do mesmo.

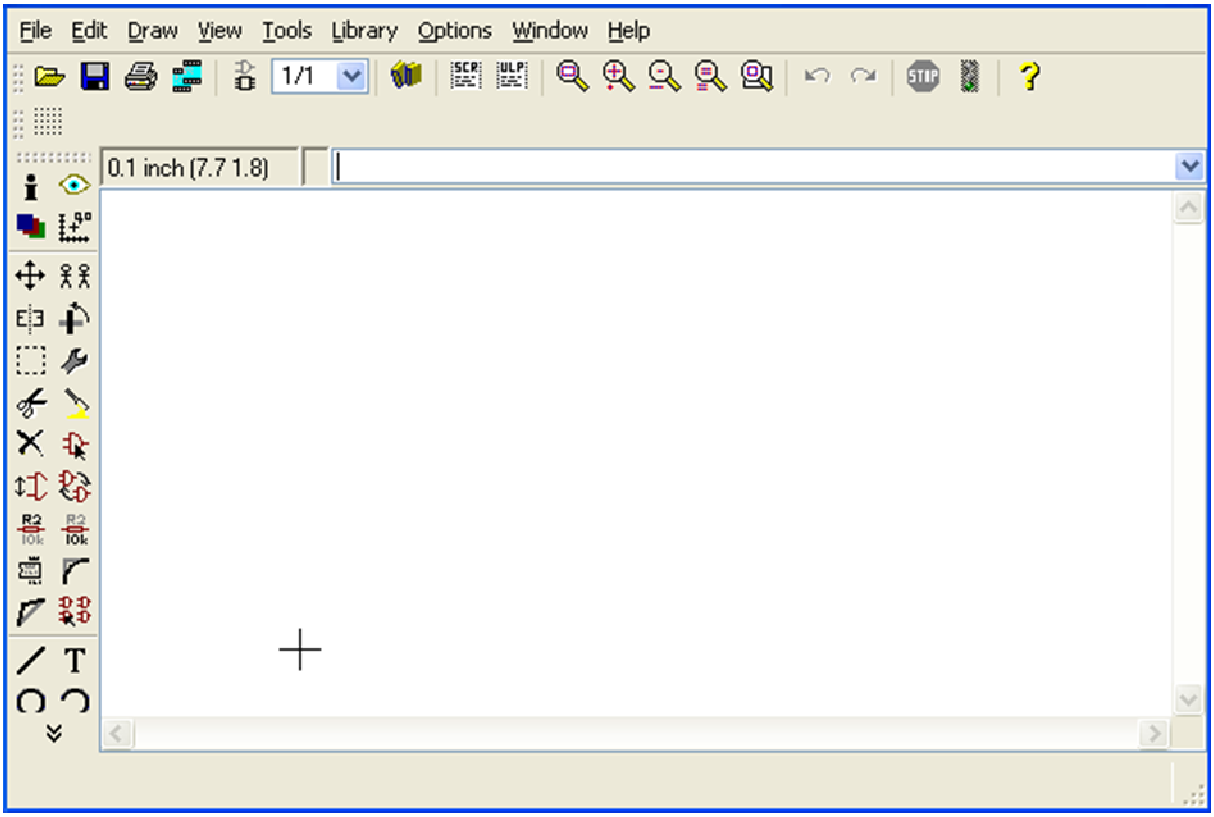


Figura 3.3 Print Screen do ambiente de construção do Esquemático, Software Eagle

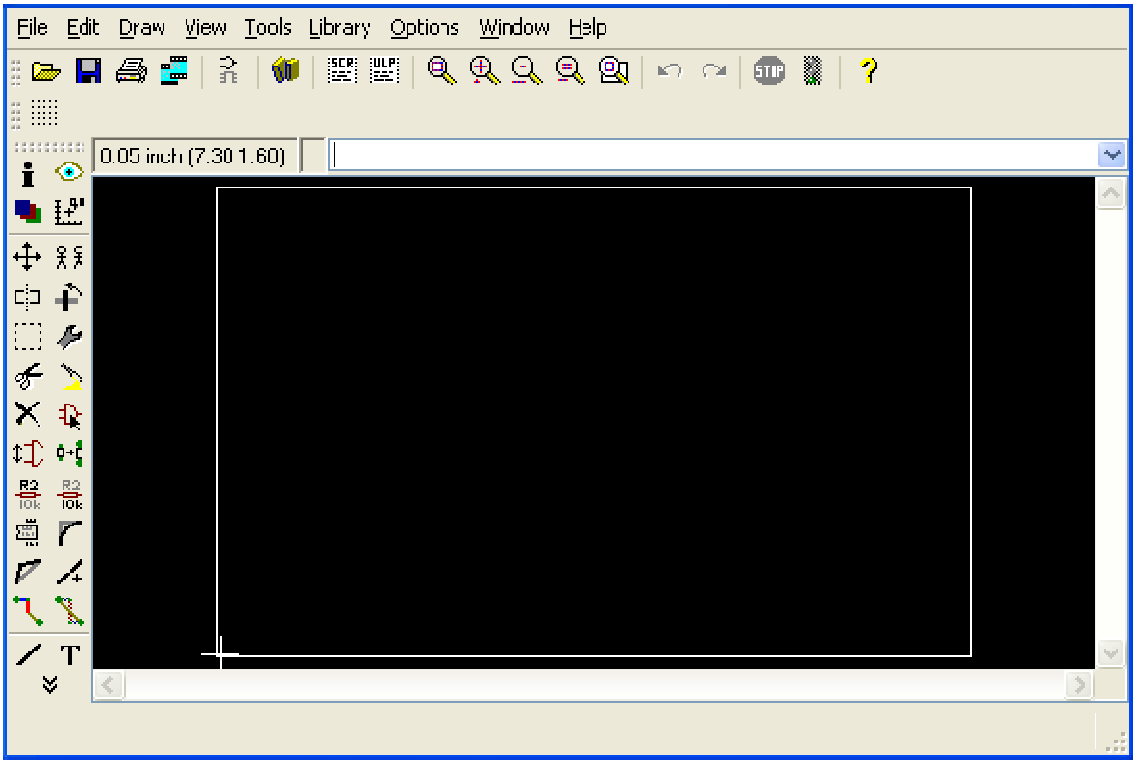


Figura 3.4 *Print Screen* do ambiente de construção do l*ayout*, Software Eagle

3.2.6. Service Author/Device Builder

Para o desenvolvimento em C# do protocolo de comunicação em rede local UPnP, utilizou-se primeiramente o *Service Author* (Figura 3.5), disponibilizado gratuitamente pela Intel, com a definição das variáveis e dos serviços que virão a ser oferecidos e consumidos. Nesse software, já se define quais e quantos serviços serão oferecidos na rede local.

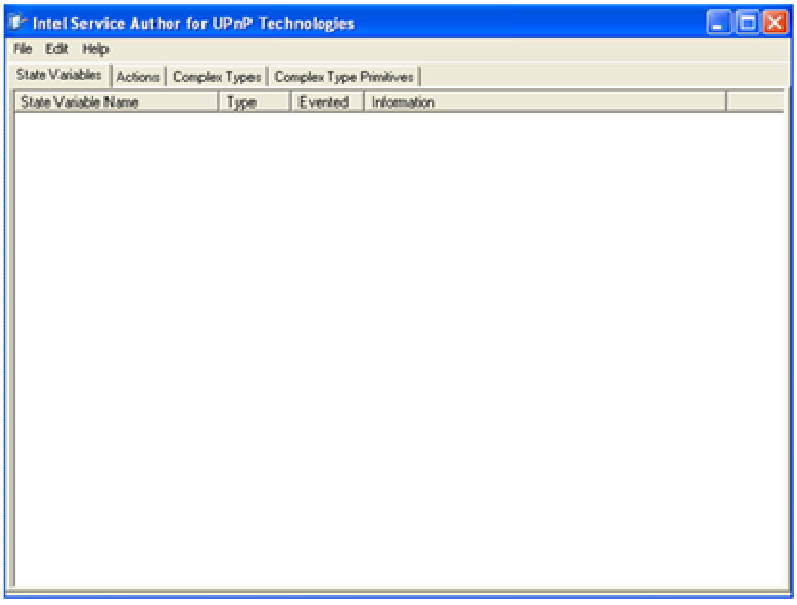


Figura 3.5 *Print Screen* do software *Service Author*

Já no software *Device Builder* (Figura 3.6), também fornecido pela Intel gratuitamente, importa-se o XML originado do *Service Author* (com a definição dos serviços), e gera-se o *Device Stack* e o *Control Point*. O *Device Stack* funciona como um servidor, ou seja, ele oferece os serviços, já o *Control Point* funciona como um cliente, requisitando e consumindo esses serviços. Os clientes podem existir em quantidade ilimitada dentro da rede local, enquanto só há um servidor.

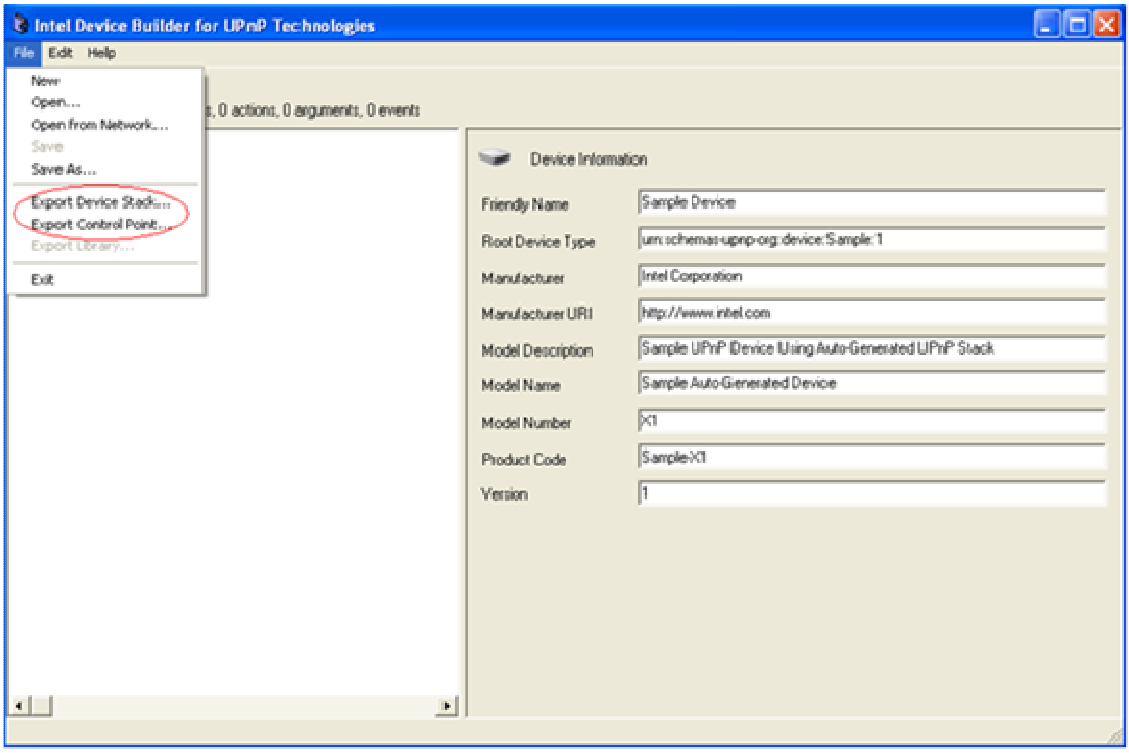


Figura 3.6 Gerando o *Device Stack* e o *Control Point*, software *Device Builder*

O *Device Builder* oferece algumas opções de ambiente para gerar o *Device Stack* e o *Control Point*, conforme Figura 3.7. Nesse trabalho, optou-se pelo Intel .NET *Framework* Stack (C#) por ser uma linguagem relativamente fácil de usar, robusta e com alta performance. Como faz parte do *framework* .NET, permite ainda a integração com ASP.NET, uma plataforma da Microsoft para o desenvolvimento de aplicações WEB que será útil no projeto.

Como ambos foram gerados a partir do mesmo XML produzido pelo *Service Author*, os *Control Points* apenas acessam os serviços providos pelo *Device Stack*. Isso aumenta a segurança, já que o *Device Stack* não reconhecerá *Control Points* criados a partir de outro XML, e não disponibilizará seus serviços.

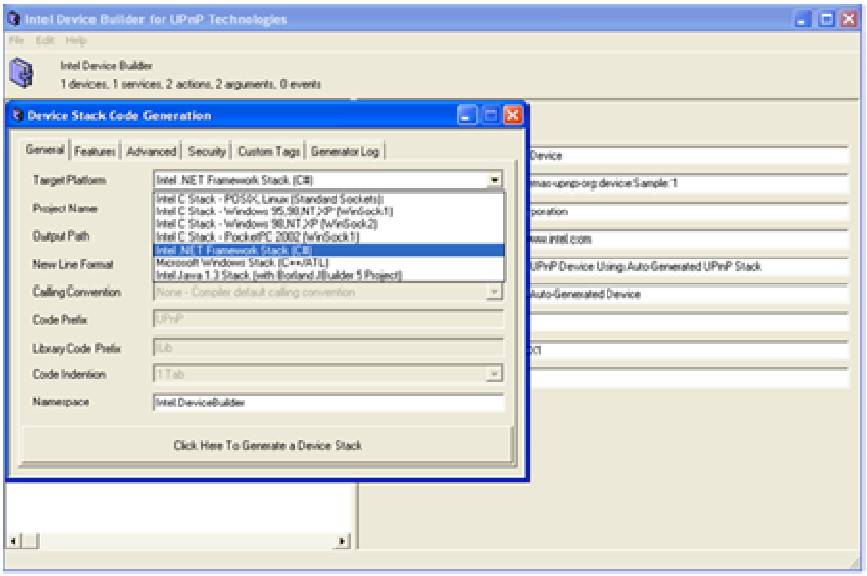


Figura 3.7 Opções de ambiente para gerar o *Device Stack*, software *Device Builder*

A partir do código gerado pelo *Device Builder*, que define a interface e realiza a comunicação entre o *Device Stack* e os *Control Points*, pôde-se então implementar as aplicações de acordo com os objetivos do projeto.

3.3. Módulos de Hardware

Esta seção descreve como estão organizados os módulos de hardware do sistema, descrevendo o funcionamento e processo de fabricação dos mesmos.

3.3.1. Diagrama dos módulos

O diagrama representando os módulos de hardware da arquitetura do sistema está ilustrado na Figura 3.8:

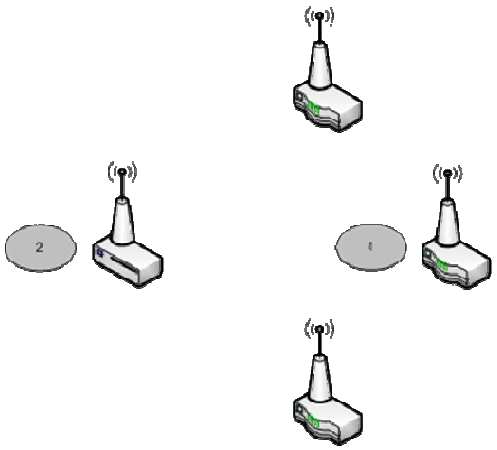
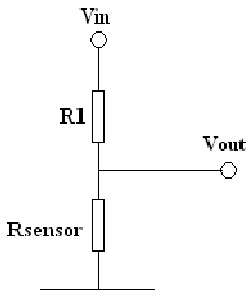


Figura 3.8 Diagrama dos módulos de hardware

A figura acima mostra três módulos Sensores de Temperatura sem fios (1) distribuídos no ambiente. Os sensores detectam a temperatura local e a transmitem para o módulo Gerenciador de Sensores (2). Todos os módulos de hardware são compostos por XBees Pro.

3.3.2. Leitura dos sensores

Os sensores térmicos [1] instalados em cada módulo Sensor de Temperatura têm dois terminais que funcionam como um resistor. Variando a temperatura no ambiente, o valor dessa resistência também varia proporcionalmente. Como nenhum dispositivo eletrônico (inclusive o XBee Pro) tem entrada para resistência, foi preciso implementar um divisor de tensão[15] (ver Figura 3.9). Com a tensão variando conforme a resistência, pode-se então adquirir um valor digital coerente através de um conversor analógico/digital, que já é disponibilizado pelo XBee Pro.



Equação 3.1 Fórmula para cálculo do divisor de tensão

$$Vout = \frac{Rsensor}{R1 + Rsensor} \cdot Vin$$

Figura 3.9 Divisor de Tensão

Vin é a tensão de alimentação da placa e **R1** é um resistor de valor fixo, logo eles não variam. Percebe-se então que a tensão de saída **Vout** (que entra no conversor A/D do XBee Pro) apenas varia com a resistência de **Rsensor**, conforme Equação 3.1.

Um conversor A/D é um circuito que converte um nível de tensão em um valor numérico (digital) correspondente. A precisão do conversor A/D do XBee Pro é de 10 bits, logo podemos obter 2^{10} (ou 1024) valores possíveis. Esse número digitalizado posteriormente passará por uma tabela para convertê-lo em um valor real de temperatura.

O XBee Pro tem cinco canais (ou pinos) que podem realizar a conversão analógica/digital, dependendo apenas da programação via X-CTU. No caso desse projeto, precisou-se apenas utilizar um dos canais, já que em cada módulo Sensor de Temperatura temos apenas um sensor. O pino escolhido foi o 20 (AD0), conforme Figura 3.10:

[15] Divisor de tensão – Forma de fazer a tensão de saída alternar de acordo com a variação de algum dos resistores ou da tensão de entrada.

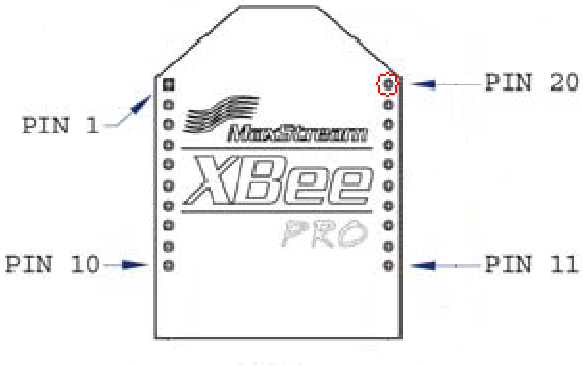


Figura 3.10 Pino de utilização do conversor A/D do XBee Pro

3.3.3. Descrição dos módulos

A Tabela 3.1 descreve os módulos de hardware, bem como suas responsabilidades:

Tabela 3.1 Descrição dos módulos de hardware

 Sensor de Temperatura	Módulo *ZigBee* que envia para o Gerenciador de Sensores o valor lido de seu canal A/D, onde se encontra o sensor.
 Gerenciador de Sensores	O Gerenciador de Sensores é o responsável pelo recolhimento dos valores de todos os Sensores de Temperatura espalhados no ambiente, uma a cada vez.

Todos os Sensores de Temperatura lêem seu canal A/D (leitura do sensor) e retransmitem os valores dessa conversão para o Gerenciador de Sensores, um a cada vez. O Gerenciador de Sensores, por sua vez, recolhe esses dados e repassa para um computador que tratará essas informações, disponibilizando-as via UPnP.

3.3.4. Processo de Fabricação

Com o protocolo de comunicação definido entre os Sensores de Temperatura e o Gerenciador de Sensores, pôde-se então prototipar os módulos XBees Pro com seus respectivos sensores de leitura. A prototipagem apenas ocorreu para os Sensores de Temperatura, já que para o Gerenciador de Sensores se utilizou a placa CON-USBBEE, descrita na subseção 3.2.4.

As fases para a construção dos Sensores de Temperatura foram as seguintes:

1 – Definição do esquemático;

2 – Confecção do *layout*;

3 – Montagem da placa.

3.3.4.1. Definição do Esquemático

Afim de não encontrar qualquer erro no final do processo de fabricação, o que acarretaria um atraso e tornaria o custo do projeto maior, todos os testes preliminares para validar o esquema foram realizados em uma *protoboard*. Esse esquema vale para todos os Sensores de Temperatura (por exemplo, Sensor de Temperatura 1, Sensor de Temperatura 2, etc.), pois a diferenciação é realizada pela programação de cada XBee Pro através do software X-CTU, descrita na subseção 3.2.4.

A partir da validação, o passo seguinte foi definir o esquemático utilizando o software Eagle, versão 4.16. A Figura 3.11 mostra o circuito com todas as conexões necessárias para o funcionamento correto de cada Sensor de Temperatura. Um esquema de proteção foi montado com o componente eletrônico LM1117 para o XBee Pro não receber uma tensão acima de seu limite de 3.3V, que certamente danificaria o dispositivo.

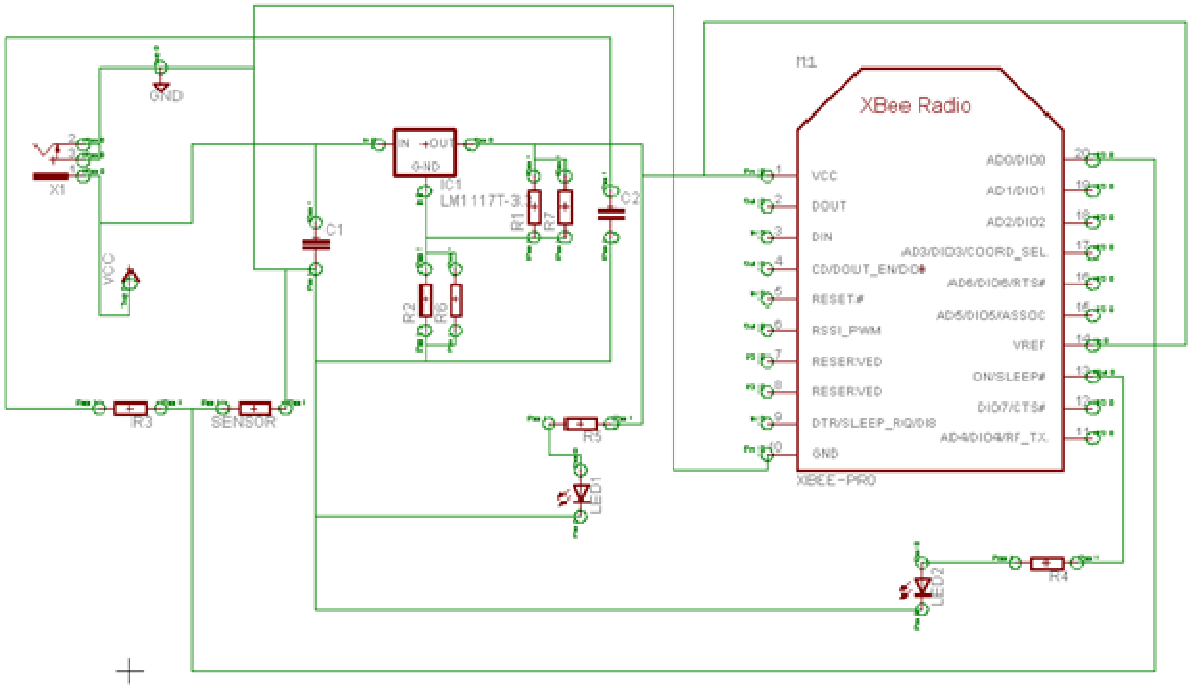


Figura 3.11 Esquema do Sensor de Temperatura

3.3.4.2. Confecção do Layout

Utilizando o mesmo software e baseando-se no esquema já criado, o layout da placa de circuito impresso foi confeccionado. Os componentes foram arranjados de forma a economizar o maior espaço possível, minimizando assim o tamanho da placa. Também foram distribuídos de tal maneira que não foi preciso utilizar nenhum *jump*[16].

O modelo geral do *layout* da placa, mostrado na Figura 3.12, representa os componentes e seus respectivos nomes, as trilhas e os orifícios necessários para a montagem. Tudo em cores diferentes para facilitar o entendimento.

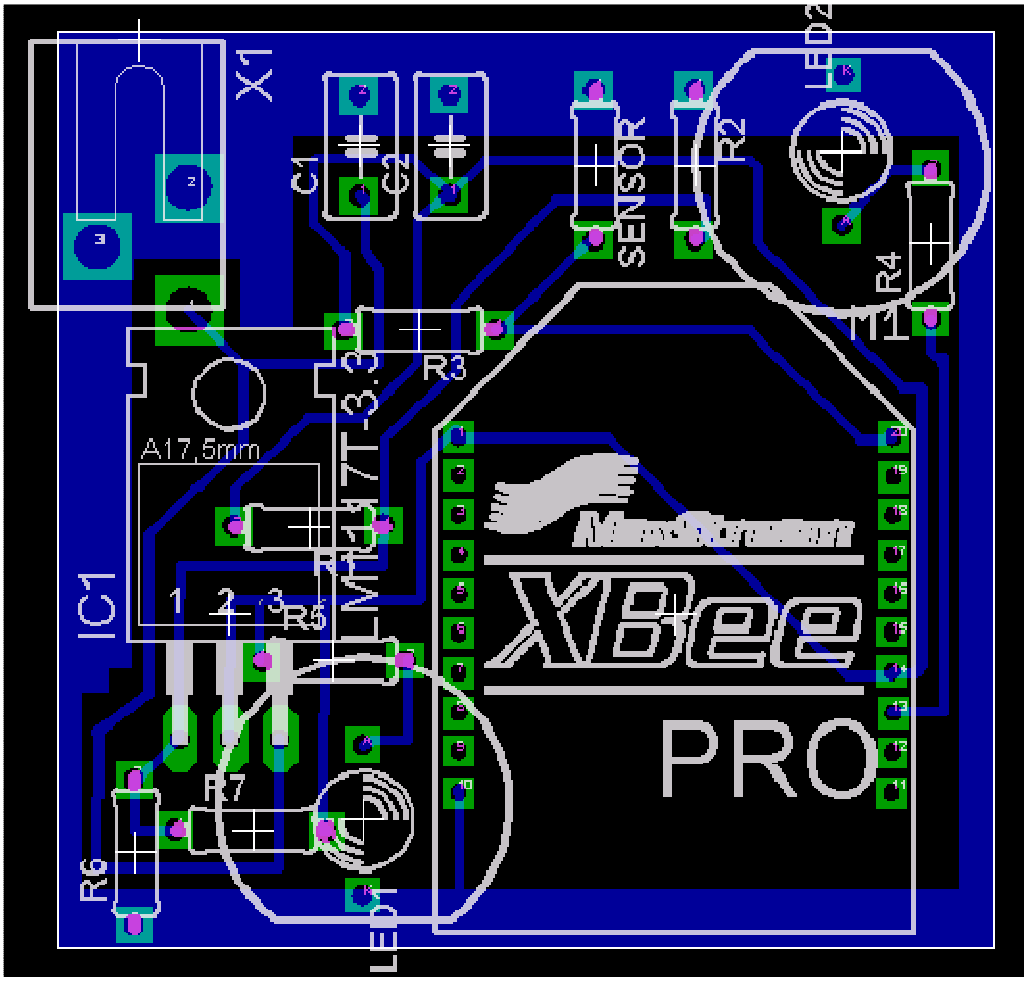


Figura 3.12 Modelo geral do *layout* da placa Sensor de Temperatura

O modelo para impressão é menos detalhado, não contendo as informações nem as representações de cada componente, conforme Figura 3.13. Esse é o modelo ideal para a fabricação da placa, com duas cores. O branco representa as trilhas e os orifícios, enquanto o preto é apenas para fazer o contraste, representando o fundo da imagem.

[16] *Jump* - Maneira de fazer uma trilha passar por cima da outra sem que haja curto-circuito.

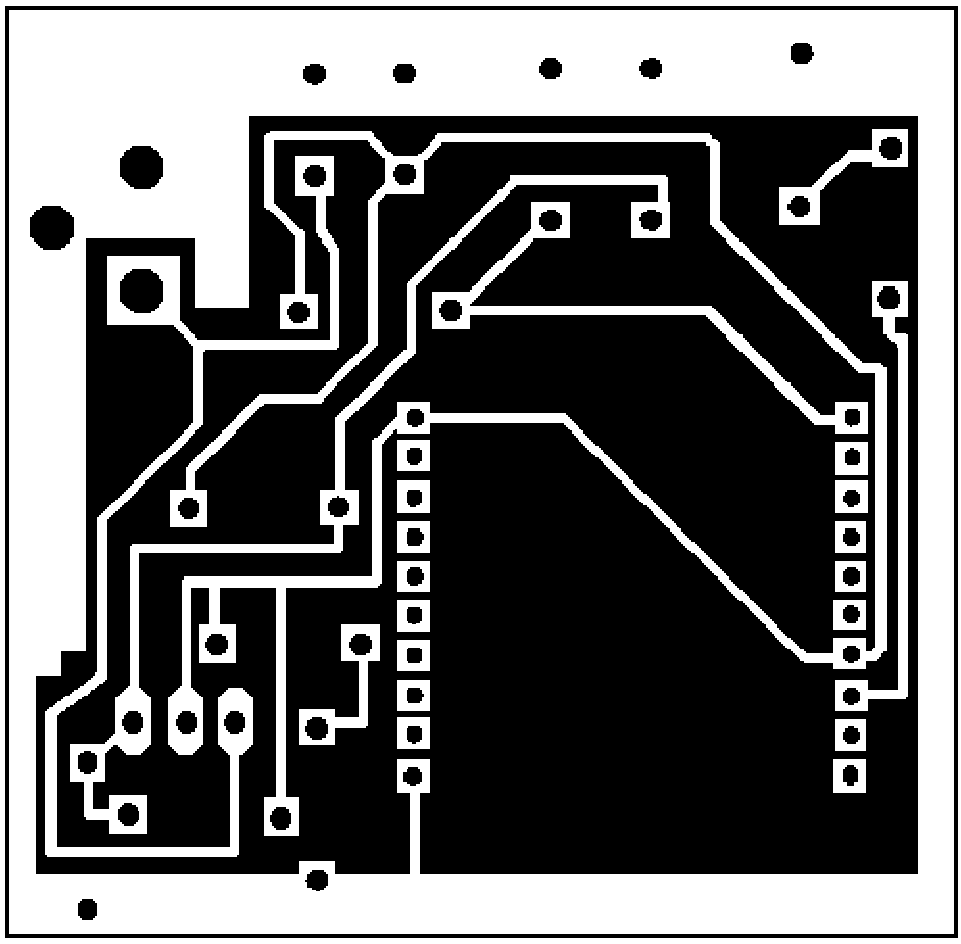


Figura 3.13 Modelo para impressão do *layout* da placa Sensor de Temperatura

3.3.4.3. Montagem da placa

Para a construção da placa, ocorreram os seguintes passos:

1 – Impressão do *layout* em um papel especial, que realça e oferece mais definição à figura.

2 – Com um ferro de passar, transferir o desenho impresso no papel para uma placa de fenolite. No caso, sai apenas as trilhas e os orifícios, que posteriormente se tornarão cobre. Esse processo varia entre três e cinco minutos.

3 – Mergulho da placa em uma solução de per cloreto de ferro com água, ocorrendo assim uma reação química. Após a corrosão, as trilhas e os orifícios ficam em cobre, finalizando assim o processo de fabricação da placa. Esse processo dura cerca de 40 minutos.

4 – Após esse processo, pode-se então perfurar os orifícios da placa de circuito impresso para a soldagem dos componentes eletrônicos. Para não ocorrer a oxidação da mesma, pinta-se com verniz.

O resultado final do processo de fabricação da placa, juntamente com a montagem, está ilustrado na Figura 3.14 e na Figura 3.15, exibidas abaixo:

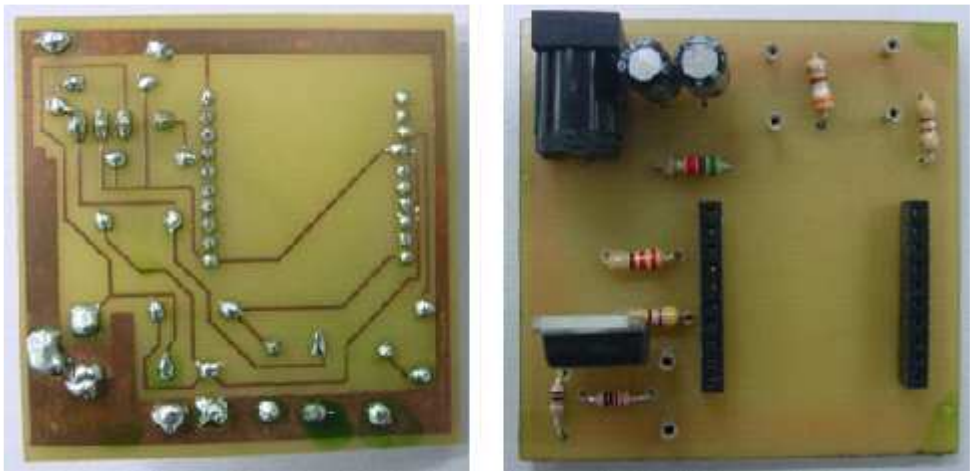


Figura 3.14 Visão inferior da placa finalizada (à esquerda); Visão superior da placa finalizada (à direita)

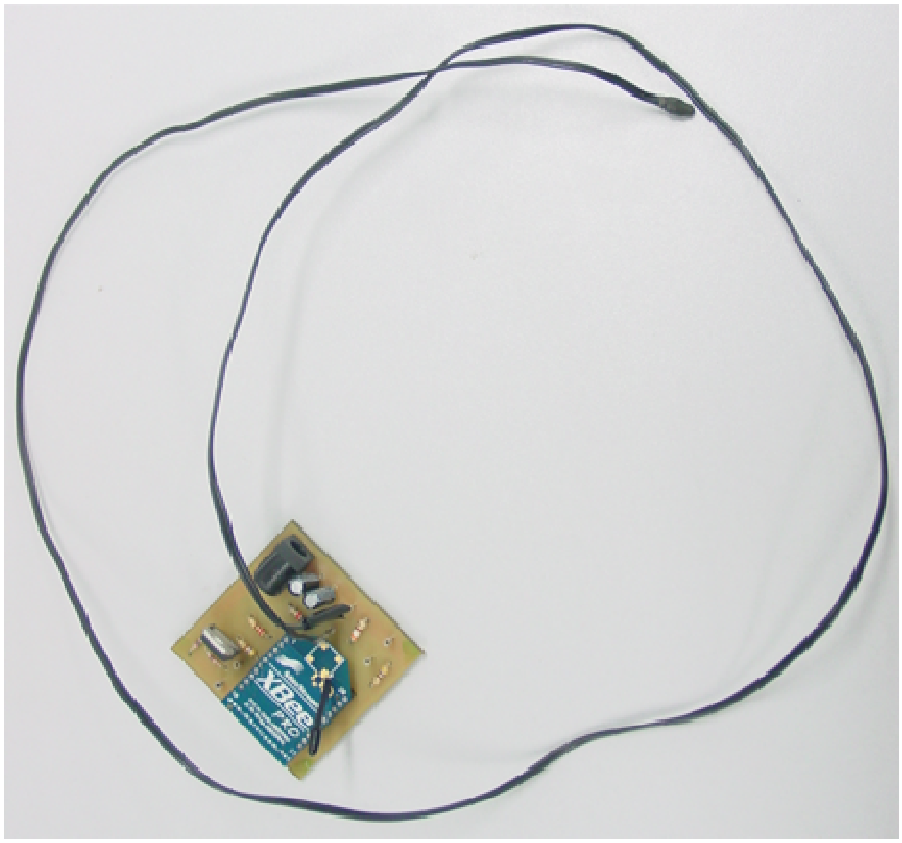


Figura 3.15 Visão superior da placa finalizada Sensor de Temperatura

3.4. Módulos de Software

Esta seção descreve como estão organizados os módulos de software do sistema, descrevendo o funcionamento dos mesmos, e como é realizada a integração com a parte de hardware.

3.4.1. Integração Hardware/Software

A integração dos módulos de software com os módulos de hardware ocorre através da comunicação entre o Gerenciador de Sensores (2) e o computador de nome Dispositivo UPnP (3). O dispositivo (2), que se encontra na placa CON-USBBEE, fica diretamente conectado e alimentado pelo PC, conforme ilustrado na Figura 3.16.

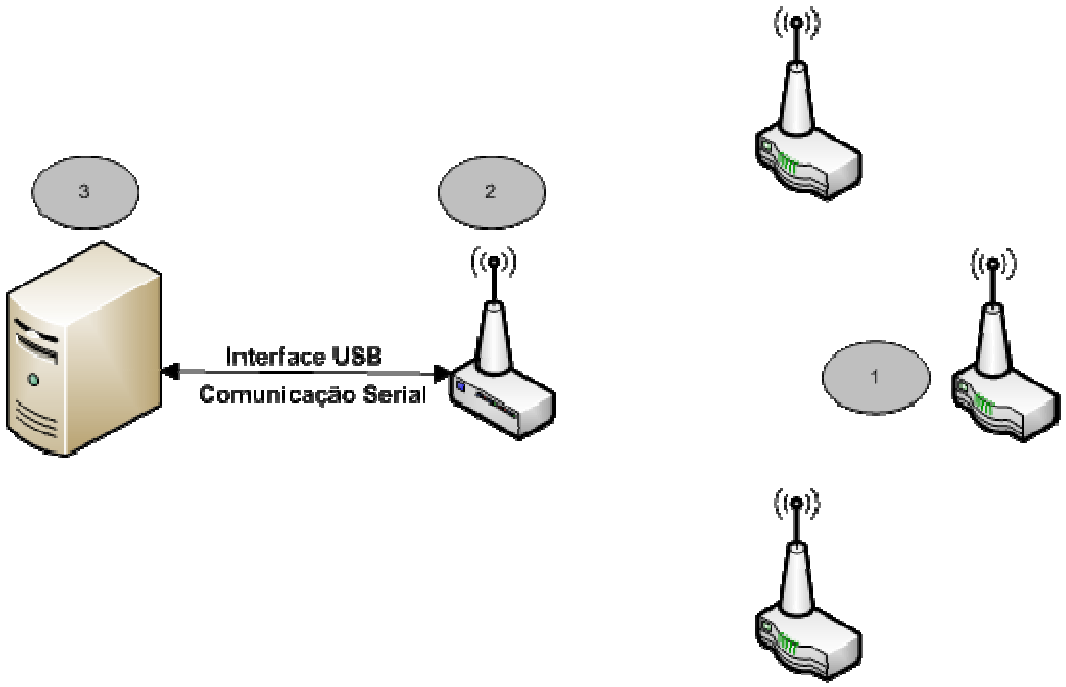


Figura 3.16 Integração do módulo de software com o módulo de hardware

Assim que recebe os dados de cada Sensor de Temperatura (1), o Gerenciador de Sensores os retransmite para a aplicação servidora no computador através da porta serial. Esses dados são recolhidos pelo Dispositivo UPnP (1), e disponibilizados através de serviço na rede local, para que os outros módulos de software possam consumí-los.

3.4.2. Diagrama dos módulos

Todos os módulos de software estão distribuídos, conectados pela rede local. O diagrama representando os módulos de software da arquitetura do sistema está ilustrado na Figura 3.17:

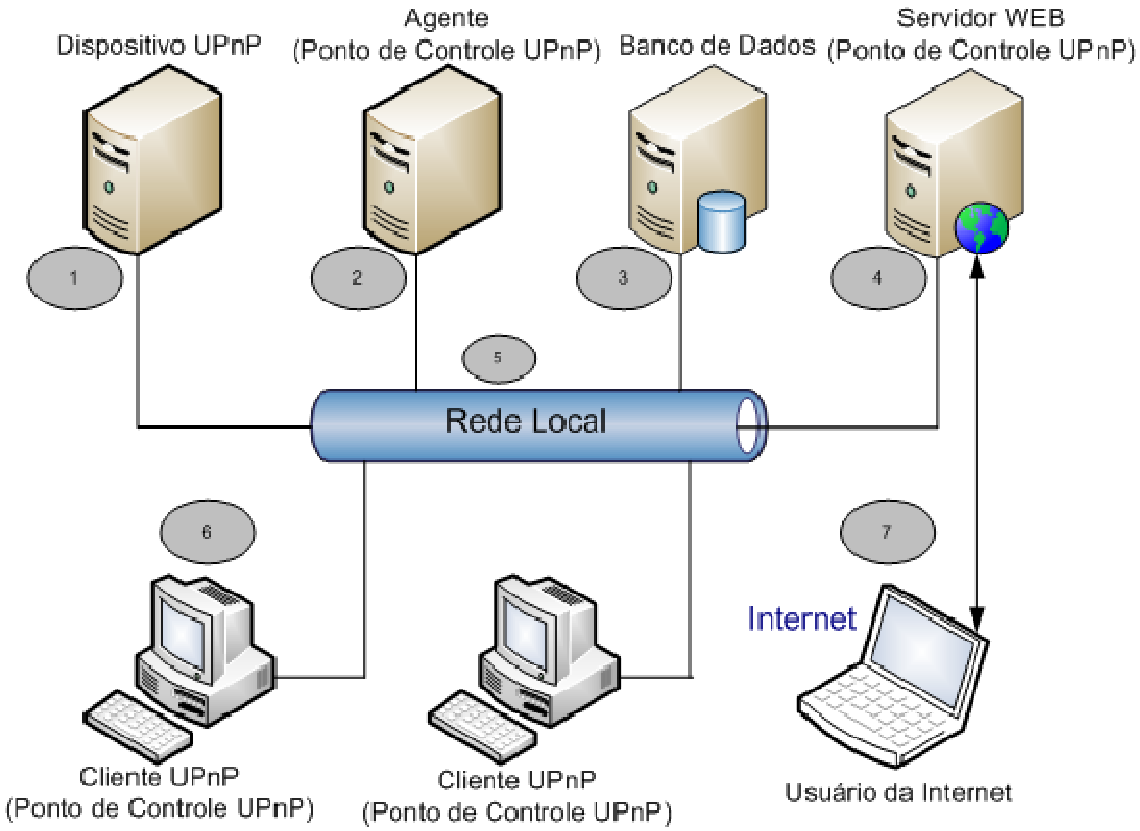


Figura 3.17 Diagrama dos módulos de software

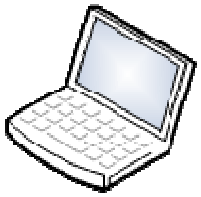
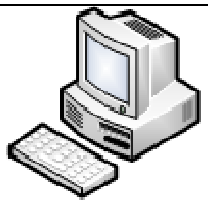
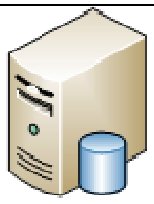
Assim que recebe o pacote de cada Sensor de Temperatura através do Gerenciador de Sensores, o Dispositivo UPnP (1) converte o valor bruto lido do canal A/D e converte para o valor real de temperatura por uma tabela. Essas temperaturas são disponibilizadas na rede local, representado por (5) para que os pontos de controle possam acessá-las. O Agente (2) tem como principal função requisitar as temperaturas e inserí-las no banco de dados (3), para que posteriormente se possa montar um gráfico de variação das mesmas. Também é responsável pela checagem no banco de dados (3) dos alarmes cadastrados, caso alguma temperatura recebida por (1) não esteja de acordo, envia-se uma mensagem de aviso para a(s) pessoa(s) cadastrada(s).

O Servidor WEB (4) é a máquina responsável pelo container IIS, que mantém as aplicações ASP.NET. São vários os serviços oferecidos por essa aplicação, como o cadastro de pessoas, de sensores e de alarmes, além de exibir um gráfico com as variações das temperaturas de cada cliente. O propósito do banco de dados (3) é justamente armazenar as informações das pessoas, sensores e alarmes cadastrados por (4), bem como as temperaturas inseridas por (2). Os Clientes UPnP, representados por (6), executam um software específico para acessar as temperaturas disponibilizadas por (1). Já o usuário da internet (7), em qualquer lugar do mundo, se tiver permissão para logar no site pode acessar todos os serviços oferecidos pela aplicação WEB, mantida por (4).

3.4.3. Descrição dos módulos

A Tabela 3.2 descreve todos os módulos de software, bem como suas responsabilidades.

Tabela 3.2 Descrição dos módulos de software

 Dispositivo UPnP (Servidor)	Responsável pela comunicação com o Gerenciador de Sensores (hardware) pela porta serial. O valor lido é convertido para o valor de temperatura real de acordo com uma tabela. Disponibiliza as temperaturas em rede local através do conjunto de protocolos UPnP.
 Agente	Consiste em um Windows *Service*[17]. Requisita as temperaturas ao servidor e as armazena no banco de dados. Este componente também é responsável por enviar mensagens de alarme para os usuários.
 Servidor WEB	O Servidor WEB é o hardware responsável pelo container IIS. Este, por sua vez, é o mantenedor da aplicação WEB.
 Usuário da Internet	Através da internet, qualquer usuário que tenha permissão para acessar a página da aplicação pode verificar as temperaturas em cada Sensor de Temperatura e cadastrar um alarme, dentre outros serviços.
 Cliente UPnP	Software específico que funciona apenas dentro da rede local, pois requisita diretamente ao servidor as temperaturas, através do conjunto de protocolos UPnP.
 Banco de Dados	O banco de dados Microsoft SQL Server 2000 é responsável pelo armazenamento das informações de temperaturas, de alarmes e de pessoas cadastradas.

[17] Windows *Service* – Aplicação que se inicializa com o Windows e é executado em *background*, oferecendo suporte para outras aplicações e/ou executando tarefas específicas.

3.5. Diagrama geral da arquitetura

Na Figura 3.18 se encontra o diagrama geral da arquitetura:

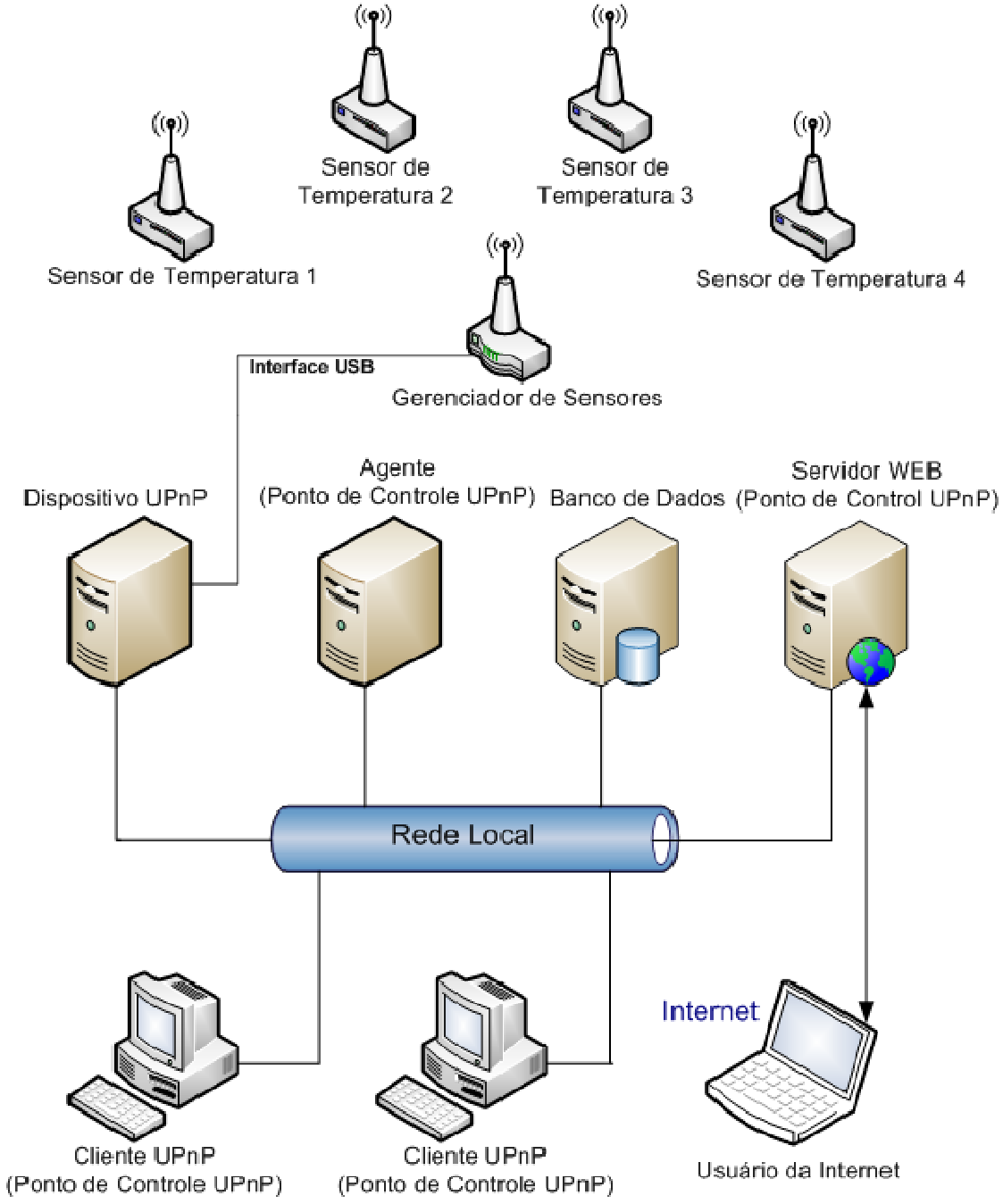


Figura 3.18 Diagrama geral do Sistema de Disponibilização de Informações Sensoriais Térmicas via UPnP

4. Implementação

Nesta seção será descrito como foi realizada a implementação referente ao Sistema de Disponibilização de Informações Sensoriais Térmicas via UPnP, desenvolvido nesse trabalho.

4.1. Hardware

Duas programações diferentes tiveram que ser realizadas nos XBees Pro, uma para o Gerenciador de Sensores e outra para os Sensores de Temperatura. As programações servem também para os dispositivos XBees, já que os dois dispositivos são compatíveis.

Antes de apresentar como foi programado os XBees Pro, é importante elucidar alguns parâmetros que podem ser configurados através do software X-CTU. Primeiro se deve definir seu modo de operação:

- **AP** (API *Enable*) – Modo de funcionamento do XBee Pro. Pode funcionar, por exemplo, como um nó coordenador que se comunica com todos os outros módulos *ZigBees*, ou também como um nó que se comunica apenas com seu coordenador;

Na área de redes e segurança, os mais importantes são:

- **MY** (*Source Address*) – Endereço fonte do módulo *ZigBee* (16 bits);

- **DL** (*Destination Address Low*) – Primeiros 32 bits (de 64 bits) do endereço destino do módulo *ZigBee*;

- **DH** (*Destination Address High*) – Os 32 bits mais significativos (de 64 bits) do endereço do módulo *ZigBee*;

- **NY** (*Node Identifier*) – Armazena o nome de identificação do dispositivo.

Em relação ao consumo de energia, estão listados abaixo os registradores utilizados:

- **SM** (*Sleep Mode*) – Define se o dispositivo entra no estado Sleep ou não;

- **ST** (*Time Before Sleep*) – Se estiver com o modo SM habilitado, define o tempo de funcionamento antes de entrar no estado *Sleep*;

- **SP** (*Cyclic Sleep Period*) - Se estiver com o modo SM habilitado, define o período entre um estado *Sleep* e o próximo.

Já com os registrados de I/O, apenas um pino foi necessário:

- **D0** (DIO0 *Configuration*) – Pino de I/O que pode funcionar de diversas formas, inclusive como conversor analógico/digital.

Existem outros parâmetros, porém, não foram de utilidade para o desenvolvimento desse trabalho de graduação. Na Figura 4.1, segue um *print screen* de um exemplo de programação pelo software X-CTU.

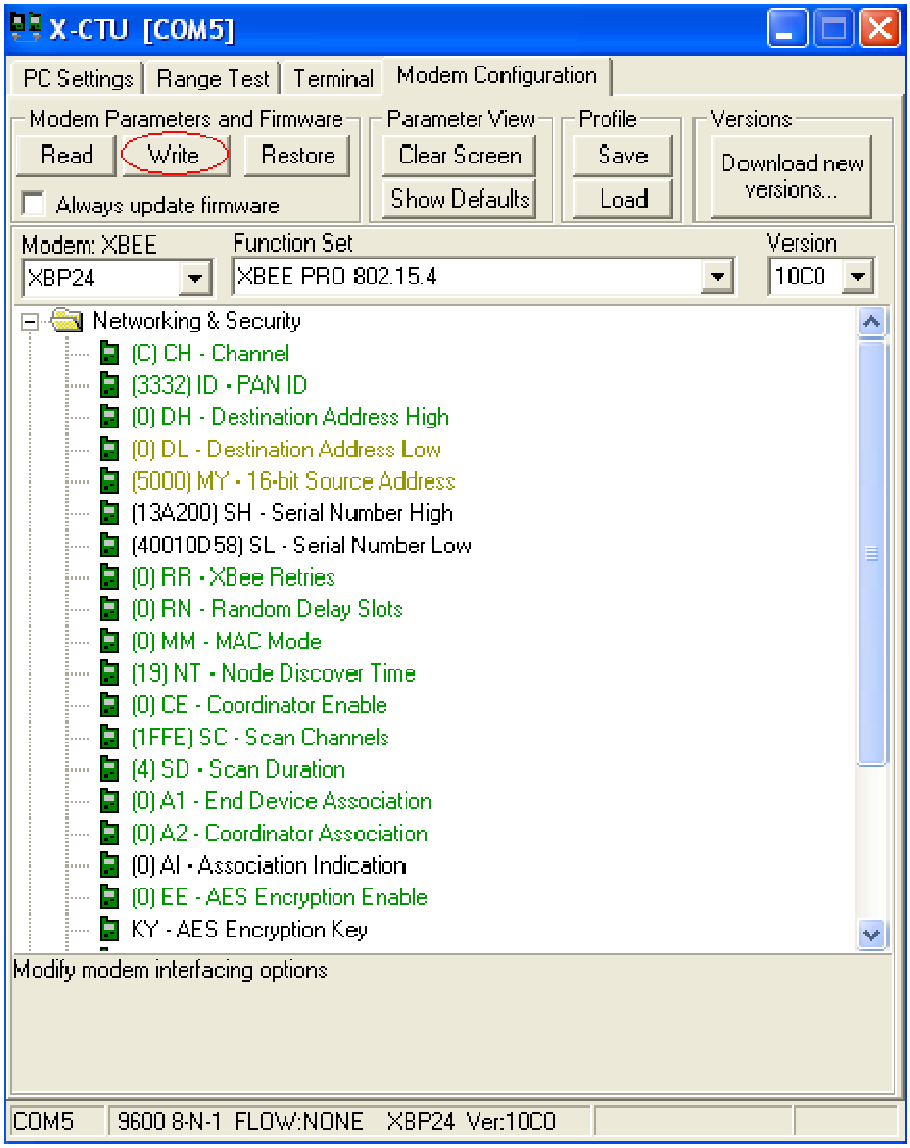


Figura 4.1 Exemplo de programação de um dispositivo XBee Pro

4.1.1. Configurando o XBee Pro do Gerenciador de Sensores

O XBee Pro do módulo Gerenciador de Sensores teve o parâmetro **AP** modificado para 0x01, habilitando o modo API. Nesse conceito, esse dispositivo funciona como um coordenador em uma rede, onde pode enviar e receber dados de todos os outros módulos *ZigBees* que estejam dentro do raio de distância tolerável, estejam trabalhando com o modo API desabilitado, e tenham como endereço destino o seu endereço fonte. O endereço fonte **MY** escolhido para o XBee Pro que atua como coordenador foi 0x5000, porém poderia ser qualquer outro número representativo em 16 bits. Com o modo API habilitado, os parâmetros **DL** e **DH** (endereço destino) perdem sua funcionalidade, pois o módulo trabalha como um coordenador e se comunica com vários outros nós, não sendo apenas um. Nesse caso, pode-se preencher esse campo com qualquer número. O **NY** utilizado para identificar o coordenador é COORD CON-USBBEE, já que ele se encontra em uma placa do tipo CON-USBBEE.

O XBee Pro coordenador não deve ser desligado em nenhum momento, pois poderia perder informações advindas de outros nós (Sensores de Temperatura), já que ele é um módulo central. Logo, o parâmetro **SM** deve permanecer em 0x00 (*Sleep Mode* desabilitado). O consumo de energia nesse caso não é problema porque o dispositivo encontra-se na placa CON-USBBEE, diretamente conectada e alimentada por um computador. Os parâmetros **ST** e **SP** ficam sem função com o **SM** desligado.

O módulo *ZigBee* coordenador não precisa utilizar nenhum pino de I/O, já que ele vai apenas receber dados por rádio freqüência vinda dos XBees Pro dos Sensores de Temperatura. Logo, o pino DIO0 deve ficar desabilitado.

Configuração do XBee Pro do módulo Gerenciador de Sensores:

- **AP = 0x01 (API *ENABLED*);**
- **MY = 0x5000;**
- **DL = XXXX;**
- **DH = XXXX;**
- **NY = COORD CON-USBBEE;**
- **SM = 0x00 (*NO SLEEP*)**;
- **ST = XXXX**;
- **SP = XXXX**;
- **D0 = 0x00 (*DISABLED*)**.

4.1.2. Configurando os XBees Pro dos Sensores de Temperatura

Os XBees Pro contidos nos módulos Sensores de Temperatura praticamente recebem a mesma configuração um do outro, mudando apenas o endereço fonte MY. Por exemplo, como foram desenvolvidos três módulos Sensores de Temperatura, cada um recebeu um MY diferente: 0x5001, para o cliente de identificação NY igual a CLIENTE1; 0x5002, para o cliente de identificação NY igual a CLIENTE2; e 0x5003, para o cliente de identificação NY igual a CLIENTE3. Com isso, evita-se o conflito e a redundância de informações. Todos devem ter o mesmo DL e DH, já que o endereço destino é um mesmo coordenador, ou seja, 0x5000 no caso desse projeto (endereço fonte do Gerenciador de Sensores). Todos devem ter também o modo API desabilitado, ou seja, parâmetro AP igual a 0x00. Isso faz com que todos os nós se comuniquem com apenas um coordenador.

Como os Sensores de Temperatura estão espalhados no ambiente, é necessária e essencial a minimização do consumo de energia dos mesmos. Para maximizar o tempo de funcionamento dos módulos, eles transmitem os dados a cada 15 segundos. No intervalo, eles entram no MODE SLEEP, com **SM** igual a 0x04. Nesse modo, eles ficam sem consumir energia em

intervalos cíclicos. Para se ter esse período de *standby* entre uma transmissão e outra, foi preciso ajustar mais dois parâmetros: **ST** igual a 0x1500 e **SP** igual a 0x200.

Para adquirir os dados do sensor de temperatura é preciso um pino de I/O. Logo o registrador **D0** foi configurado para 0x02, fazendo o pino 20 do XBee Pro operar como um conversor analógico/digital. Esse dado lido do pino e transmitido para o Gerenciador de Sensores ainda não é o valor final de temperatura, precisando passar por uma tabela de conversão no computador se comunica com esse módulo coordenador.

Configuração dos XBees Pro dos módulos Sensores de Temperatura:

- **AP = 0x00 (API *DISABLED*);**
- **MY = XXXX;**
- **DL = 0x5000;**
- **DH = 0x00;**
- **NY = CLIENTEX;**
- **SM = 0x04 (*CYCLIC SLEEP REMOTE*)**;
- **ST = 0x1500**;
- **SP = 0x200**;
- **D0 = 0x02 (ADC)**.

4.2. Software

Primeiramente será mostrado como ocorreu a integração dos módulos de software com os módulos de hardware. Os dados de controle recolhidos pelo módulo coordenador (Gerenciador de Sensores) são repassados diretamente para um computador, chamado de Dispositivo UPnP. Essa comunicação ocorre via porta serial através da placa CON-USBBEE, como dito anteriormente.

Esses dados de controle incluem:

- Delimitador Inicial, indicando que a transmissão irá começar (1 Byte);

- Tamanho do pacote (2 Bytes);

- Identificador de API, indicando qual foi a operação realizada pelo *ZigBee*. No caso desse projeto, a operação realizada por cada Sensor de Temperatura foi a conversão A/D (1 Byte);

- Endereço, identificando cada Sensor de Temperatura (2 Bytes);

- RSSI, nível de potência do sinal (1 Byte);

- Opções, status de transmissão dos dados (1 Byte);

- Cabeçalho, contendo o número total de amostras recolhido por cada Sensor de Temperatura, e quais os canais A/D e canais de dados que estão ligados (3 Bytes);

- Valor do canal A/D (2 Bytes);

- Checksum, para verificação de erro (1 Byte).

4.2.1. Dispositivo UPnP (Servidor)

Os dados de controle advindos da porta serial são convertidos através de uma tabela para um valor de temperatura real, identificando qual o Sensor de Temperatura que os enviou pelo endereço. A partir da recepção dessas informações, pode-se então disponibilizá-las na rede local através do conjunto de protocolos de rede UPnP. Por isso, esse módulo recebeu o nome de Dispositivo UPnP (aplicativo servidor), exibido na Figura 4.2.

Essa aplicação recolhe os dados junto ao Gerenciador de Sensores, converte-os em um valor real de temperatura e disponibiliza as informações na rede local, através de serviços UPnP. Foi desenvolvida a partir do código já gerado do *Device Stack* (produzido pelo software *Device Builder*), como foi detalhado na subseção 3.2.6. A implementação da aplicação ocorreu em C#, mesma linguagem do *Device Stack*, na plataforma .NET.

O Dispositivo UPnP pode disponibilizar até 16 temperaturas de Sensores de Temperatura ao mesmo tempo. Os correspondentes valores brutos, ou seja, os valores que ainda não passaram pela tabela de conversão, também são disponibilizados na rede local. Logo, são 32 serviços diferentes (de 16 Sensores de Temperatura) que podem ser requisitados a qualquer momento pelos *Control Points*. Essa quantidade foi definida já no software *Service Author*, conforme subseção 3.2.6.

Se em algum momento o sensor estiver desligado ou fora de alcance, disponibiliza-se 00 para o valor bruto e -1001 para o valor de temperatura. Não se disponibiliza 00 para temperatura porque o mesmo é um valor aceitável, enquanto -1001 indica que houve algum erro na operação de requisição.

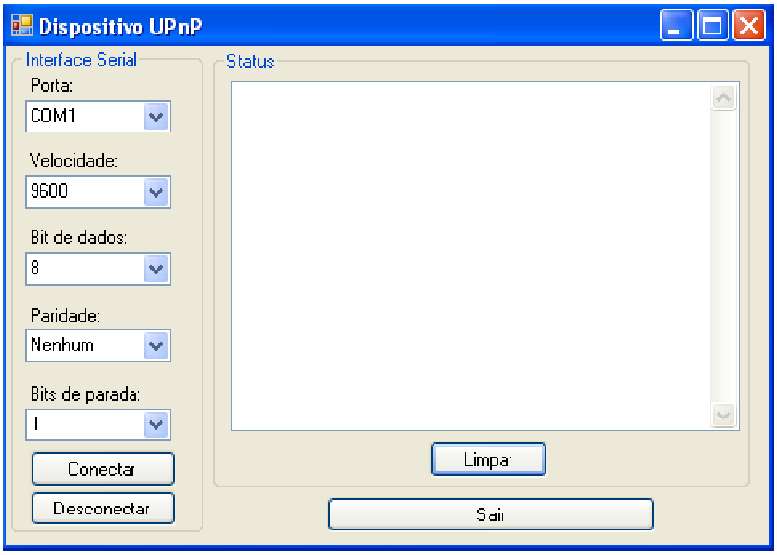


Figura 4.2 *Print Screen* da interface gráfica do Dispositivo UPnP

A interface serial da tela serve para conexão com o XBee Pro do módulo Gerenciador de Sensores. A partir dessa comunicação são adquiridos todos os dados de controle, que contém as informações dos módulos Sensores de Temperatura. Já no campo Status são exibidas todas as requisições feitas pelos *Control Points*.

Aplicações com diferentes finalidades foram desenvolvidas para acessar o Dispositivo UPnP, todas elas a partir de *Control Points* já gerados pelo *Device Builder*, vide subseção 3.2.6. Uma aplicação mais específica, para acesso direto aos valores brutos e às temperaturas, com o nome de Cliente UPnP. Já o Agente tem a função de requisitar as temperaturas e armazená-las em um banco de dados, além de verificar alarmes cadastrados pela página mantida pelo Servidor WEB. Esse último prover vários serviços, entre eles cadastro de pessoas, de alarmes e de sensores, e ainda exibe o histórico das temperaturas através de um gráfico.

4.2.2. Clientes UPnP (Ponto de Controle)

Clientes UPnP (Figura 4.3) são módulos que executam um software específico para acesso ao Dispositivo UPnP. Eles podem existir em quantidade ilimitada dentro da rede local.

Figura 4.3 *Print Screen* da interface gráfica de um Cliente UPnP

Na interface gráfica existe um botão *Scan* que se conecta ao servidor (Dispositivo UPnP), e um botão *Get*, que faz a aquisição dos valores brutos e das temperaturas de 6 Sensores de Temperatura. Poderiam ser 16, porém como só foram fabricados 3 Sensores de Temperatura, resolveu-se colocar na interface apenas 6. O campo *Informations* exibe a identificação do Sensor de Temperatura.

4.2.3. Agente UPnP (Ponto de Controle)

O Agente UPnP é um Windows *Service* criado na plataforma .NET, que tem como função requisitar as temperaturas ao Dispositivo UPnP e inserir no banco de dados. É importante guardar as temperaturas para manter o histórico. O agente também tem a missão de checar se algum alarme deve ser disparado. A comunicação, tanto com o Dispositivo UPnP quanto com o banco de dados, é realizada através da rede local. Como ele executa em *background*, esse aplicativo não tem interface gráfica.

O agente armazena apenas as temperaturas das últimas 24 horas, para não sobrecarregar o BD. No primeiro dia de funcionamento, ele insere sem nenhuma verificação. A partir do segundo dia, ele apenas insere depois de remover a temperatura mais antiga, funcionando como uma fila.

A cada 60 segundos, o Agente UPnP requisita ao servidor as temperaturas de todos os Sensores de Temperatura ligados. Logo depois, ele faz a inserção no banco de dados, se for no primeiro dia; ou faz a remoção da última temperatura seguida de uma inserção, se for a partir do segundo dia. Esse último é o caso mais geral, pois uma vez ligado o computador, ele não mais será desligado.

Se o Sensor de Temperatura estiver ligado, ele armazena a temperatura real no banco de dados, juntamente com sua identificação. Se não, ele insere um valor de 5000, que funciona como um código apenas para guardar o tempo que aquele sensor está desligado. Isso será importante na hora de plotar o gráfico na página mantida pelo Servidor WEB, como será explicado na seção seguinte.

Outra funcionalidade do Agente UPnP é analisar se algum alarme precisa ser disparado. Os alarmes são cadastrados também pela página que se encontra no Servidor WEB. A cada 15 segundos o agente faz uma requisição ao Dispositivo UPnP, e em seguida faz uma checagem com todos os alarmes cadastrados no banco de dados. Se a temperatura estiver fora do limite, envia-se então no mesmo momento um email para a(s) pessoa(s) cadastrada(s) para recebê-lo.

4.2.4. Servidor WEB (Ponto de Controle)

O Servidor WEB tem como objetivo hospedar uma página com várias funcionalidades. Essa página foi desenvolvida em ASP.NET, como mencionado na subseção 3.1.3. Para acessar o Dispositivo UPnP foi necessário a criação de uma DLL[18], que também foi desenvolvida a partir de um *Control Point*. Com a DLL adicionada na aplicação WEB ou em qualquer sistema na rede local, torna-se possível realizar as requisições de temperatura ao servidor.

Essas páginas (da aplicação WEB) também têm a função de acessar o banco de dados fazendo as operações básicas de cadastramento, edição e remoção de pessoas, de sensores e de alarmes. A comunicação dá-se através da rede local. Tudo isso é importante por oferecer ao usuário o poder gerenciar todos os sensores distribuídos em um ambiente, independente de onde esteja, já que todas as operações são realizadas via internet.

[18] DLL (*Dynamically Linked Library*) - Conjunto de funções e rotinas de programação que podem ser acessadas dinamicamente por um programa.

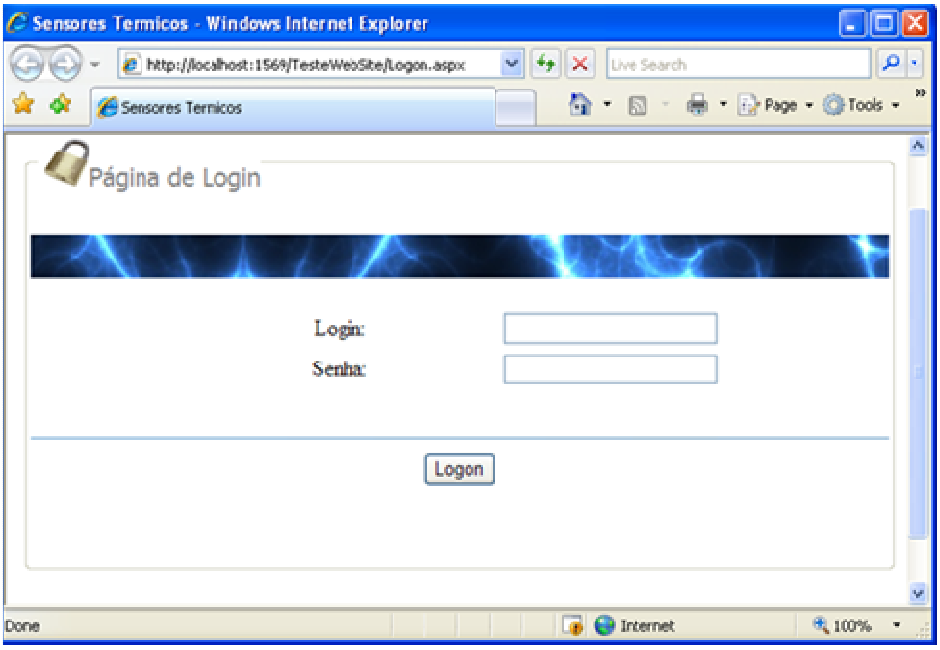


Figura 4.4 Tela de Login da aplicação WEB

A aplicação WEB trabalha com dois perfis de usuários, Gerentes e Usuários Comuns. Gerentes têm acesso irrestrito, podendo acessar todas as funcionalidades da página. Já os Usuários comuns não têm a permissão de entrar na página, apenas são inseridos para receber notificações de alarme cadastrado por algum Gerente. A Figura 4.4 mostra a tela de Login da aplicação, enquanto a Figura 4.5 exibe a tela de escolha para cadastramento de Gerente ou de Usuário Comum.

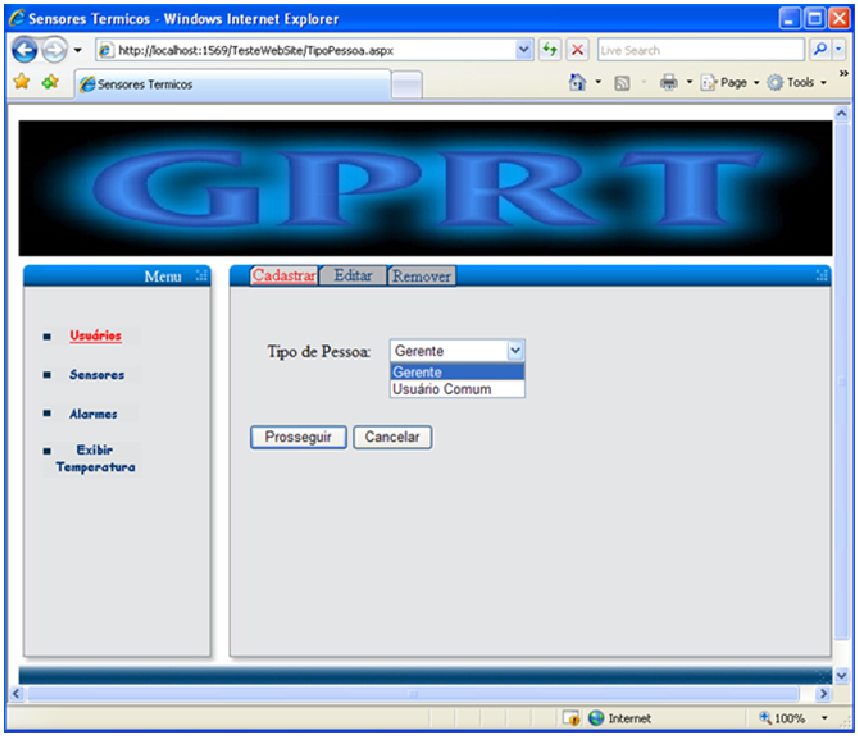


Figura 4.5 Tela de escolha entre Gerente e Usuário Comum para cadastramento de Pessoa

A única diferença no cadastramento de Usuário Comum e de Gerentes consiste nos campos Login, Senha e Confirma Senha. Eles servem para autenticação na tela de Login, pois apenas os Gerentes podem ter acesso à página. A Figura 4.6 exibe a tela de cadastramento de Gerente.

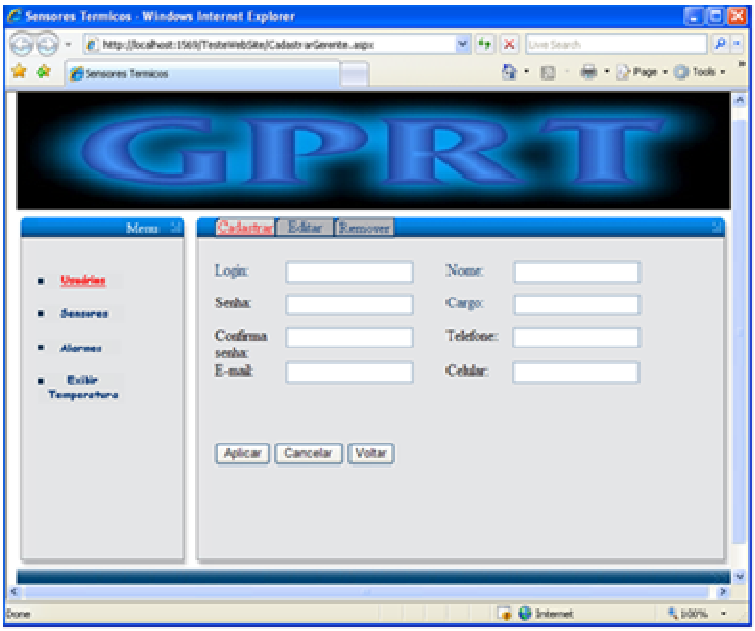


Figura 4.6 Tela de Cadastramento de Gerente

Na área de Sensores (ver Figura 4.7), pode-se apenas editá-los e removê-los. Como existe uma quantidade limite Sensores de Temperatura, não se tornou preciso criar uma aba para cadastramento, sendo inseridos inicialmente no banco de dados. Na parte de edição é possível armazenar o lugar em que o sensor está e a descrição do mesmo.

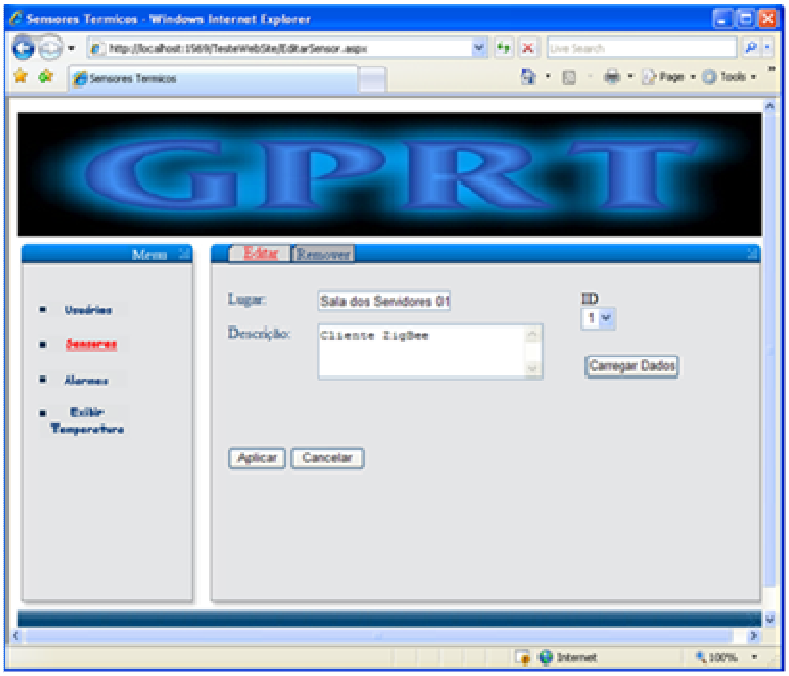


Figura 4.7 Tela de edição de Sensores

Já na parte de Alarmes, conforme Figura 4.8, as três operações básicas podem ser realizadas: cadastrar, editar e remover. Na tela de cadastro, escolhe-se uma combinação de temperatura máxima e mínima aceitável para um ou mais sensores. Coloca-se ainda uma ou mais pessoas como responsáveis por aquele alarme. Ou seja, se o Agente UPnP verificar que a temperatura lida no momento está fora do cadastrado, ele imediatamente envia emails para as pessoas responsáveis com as informações do alarme, como o nome do mesmo, o lugar em que o sensor se encontra, a temperatura em que se encontra, entre outras.

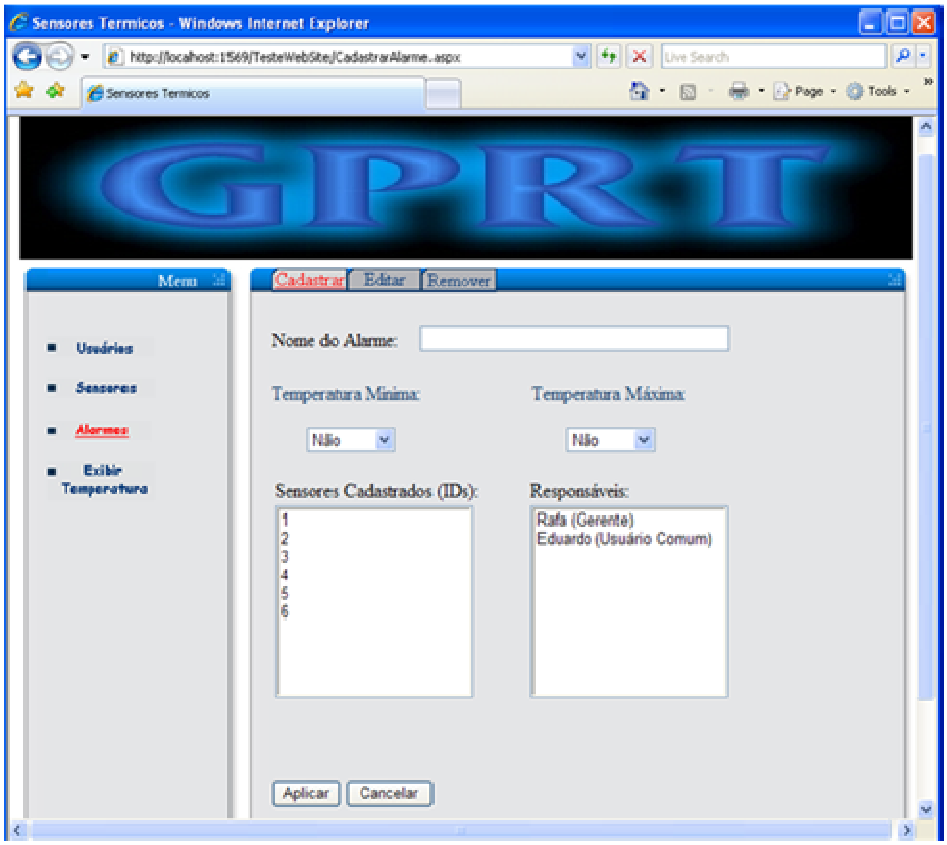


Figura 4.8 Tela de cadastramento de Alarmes

Na seção de Exibir Temperatura, ao clicar no *link* abre-se uma nova janela com a temperatura em tempo de real de cada Sensor de Temperatura, ou seja, em qualquer do mundo pode-se ter acesso às mesmas (ver Figura 4.9). Sempre que a página é atualizada, o que ocorre automaticamente a cada 10 segundos, a aplicação WEB (através da Dll criada) faz uma requisição para o Dispositivo UPnP de todas as temperaturas, e as exibe na tela.

Além disso, esta aplicação é responsável por adquirir junto ao banco de dados todas as temperaturas dos sensores das últimas 24 horas, armazenadas pelo Agente UPnP. A partir dessas informações exibe-se então um gráfico com o histórico de variações do último dia.

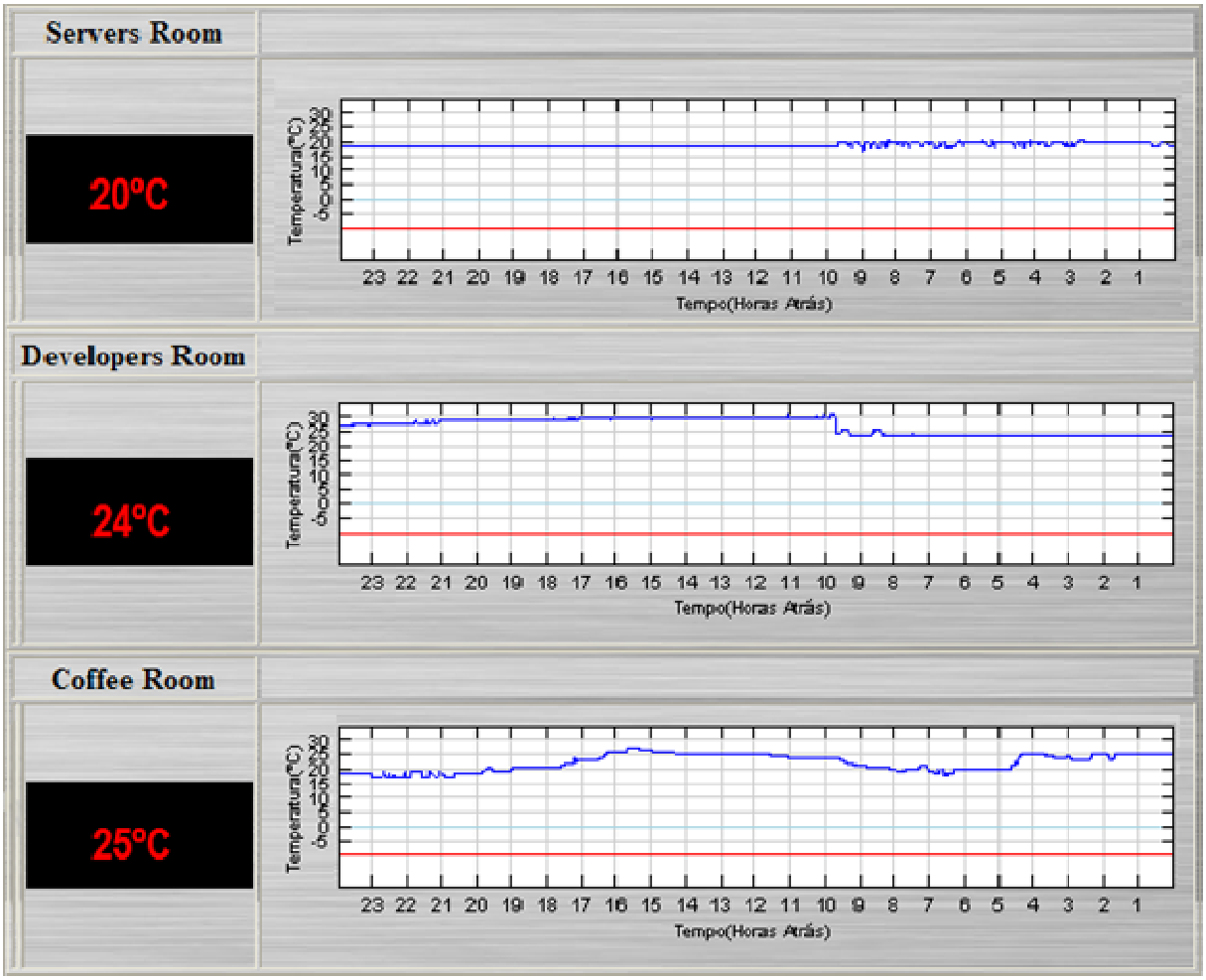


Figura 4.9 Tela de exibição da temperatura e do gráfico

4.2.5. Banco de Dados

O banco de dados Microsoft SQL Server 2000 tem a função de armazenar dados de temperatura, de usuários comuns e de gerentes, de sensores e de alarmes. Várias tabelas e *store procedures*[19] foram criadas a fim de eliminar a inconsistência das informações, além de prover um ambiente eficaz de recuperação e inserção de dados.

4.3. Experimentos e Resultados

Os testes de software foram realizados em paralelo com o desenvolvimento do mesmo. Logo, esta seção será reservada para os testes de hardware, onde características como consumo de energia e alcance entre os nós sensores serão analisados.

[19] *Store Procedures* - Programas SQL escritos por um programador e que são armazenados no servidor de banco de dados e podem ser invocados por aplicações clientes.

4.3.1. Consumo

O baixo consumo de energia é um fator determinante para o sucesso do projeto. As placas prototipadas para os módulos Sensor de Temperatura devem receber uma alimentação de no mínimo 3.3V. O módulo coordenador não necessita de atenção especial, já que como dito anteriormente este dispositivo encontra-se plugado diretamente a um computador via porta USB, sendo alimentado pelo mesmo.

Os módulos Sensor de Temperatura, que contém os dispositivos XBees Pro, são responsáveis pela leitura do canal A/D (onde se encontra o sensor térmico) e logo após pelo envio dessa informação para o módulo coordenador. Para o experimento, foi adquirida uma bateria de 9V, modelo Rayovac, conforme Figura 4.10. Para economizar energia, configurou-se o XBee Pro para coleta dessa informação a cada 15 segundos, com o imediato envio para o Gerenciador de Sensores (módulo coordenador).

Figura 4.10 Bateria utilizada para alimentar a placa confeccionada (à esquerda); Placa em funcionamento sendo alimentada por uma bateria (à direita)

Dois experimentos foram realizados na placa que corresponde ao módulo Sensor de Temperatura. Ambos os testes se deram com o sensor de resina [1].

Experimento 1

O primeiro experimento foi realizado com a configuração inicial, onde os dispositivos XBees Pro (de cada módulo Sensor de Temperatura) fazem a coleta de temperatura a cada 15 segundos e a imediatamente enviam para o módulo coordenador (Gerenciador de Sensores), conforme descrito na subseção 4.1.2.

Esse pequeno intervalo de aquisição da informação serve para deixar a temperatura o mais próximo possível do valor real. Em contrapartida, acaba gastando mais energia. A bateria Rayovac de 9V conseguiu alimentar a placa durante aproximadamente 11 horas.

Experimento 2

Já no segundo experimento, alterou-se a configuração do modo *Sleep*, com a coleta das informações passando a ser realizada a cada 4 minutos e meio. Os parâmetros **ST** e **SP** receberam novos valores, ‘0x1000’ e ‘0x68B0’ respectivamente.

Com isso a placa ganhou em economia de energia, porém perdeu em precisão, já que passa um longo período de tempo sem fazer coleta de novo dados. A bateria Rayovac de 9V alimentou a placa nessa configuração por um pouco mais de 30 horas.

4.3.2. Alcance

A máxima distância prevista para a comunicação entre módulos XBee Pro é de 1,6 quilômetros (sem obstáculos), como descrito na subseção 3.1.1.1. Um experimento foi realizado a fim de testar o real alcance máximo que permite o funcionamento correto na comunicação entre os dispositivos de hardware XBee Pro.

Mesmo com obstáculos o resultado foi bem satisfatório, funcionando perfeitamente em um raio de aproximadamente 300 metros.

4.4. Dificuldades Encontradas

Algumas dificuldades durante o desenvolvimento do projeto serão aqui relatadas:

1 – Aquisição de componentes eletrônicos: a maioria teve de ser adquirida fora do estado de Pernambuco, tendo em vista que os mesmo não foram encontrados nesta localidade. Isso acarretou um aumento no custo total do projeto

2 – Pouca experiência com hardware: dificuldades foram encontradas no momento de montar o circuito necessário para a comunicação entre os dispositivos XBee Pro. Um atraso foi ocasionado devido à grande instabilidade causada pelos testes em *protoboard*.

3 – Prototipação das placas: como foram desenvolvidas em um processo de fabricação totalmente manual e experimental, algumas placas foram refeitas devido a algumas imperfeições.

4 – Poucos tutoriais sobre como desenvolver aplicações com a tecnologia UPnP.

5 – Impossibilidade de debugar o aplicativo Agente UPnP desenvolvido durante o trabalho. Como é um Windows *Service*, o Microsoft Visual Studio .NET 2005 não oferece esse recurso para tratamento de erros, diminuindo a produtividade.

6 – Nenhuma experiência com HTML para desenvolver as aplicações WEB, contornado com muita pesquisa e estudo.

7 – Plotagem dos gráficos das últimas 24 horas de cada sensor. As bibliotecas gráficas compatíveis não ofereciam as propriedades necessárias para alcançar tal objetivo.

5. Conclusão

Neste capítulo, serão apresentados as dificuldades encontradas, as considerações finais, as principais contribuições e os possíveis trabalhos futuros relacionados ao tema abordado neste documento.

5.1. Considerações Finais

Esta monografia teve como objetivo principal propor um sistema de integração entre sensores térmicos e a disponibilização de informações. Definiu-se as tecnologias adotadas de acordo com a característica do projeto.

Para a comunicação dos sensores se utilizou o padrão IEEE 802.15.4 (*ZigBee*), responsáveis pela coleta das informações térmicas no ambiente em que estão inseridos e envio para um módulo coordenador.

A partir da recepção desses dados pode-se então disponibilizá-los em rede local através da tecnologia UPnP. Alguns aplicativos UPnP foram implementados, com as mais diversas funcionalidades. Um deles é responsável pela aquisição das temperaturas junto ao módulo coordenador e oferecimento desses serviços em rede local. Todos os outros têm a função de consumir esses serviços, fazendo a requisição das temperaturas.

Já para adquirir as informações térmicas em rede local e disponibilizar na internet foi desenvolvida uma página WEB, que tem como funções ainda cadastro de pessoas, alarmes e sensores, além da exibição de um gráfico com as temperaturas das ultimas 24 horas de cada sensor.

Alguns testes foram realizados ainda com o intuito de analisar o alcance máximo dos dispositivos responsáveis pela coleta, bem como o consumo de energia.

5.2. Principais Contribuições

Dentre as principais contribuições desse trabalho destacam-se:

- Estudo detalhado com vantagens e desvantagens sobre as tecnologias relacionadas: transmissão de dados sem fio para criação da rede de sensores; sistemas de coordenação para realização da interação entre dispositivos em rede local; e ainda as principais tecnologias de desenvolvimento WEB para difusão das informações na internet.

- Modelagem de um sistema de disponibilização via UPnP de informações sensoriais térmicas na rede local e na internet, onde a transmissão de dados sem fio ocorre através do protocolo *ZigBee*.

- Proposta de uma implementação para o sistema acima citado, incluindo o hardware e o software necessários para a criação do mesmo. Também foi descrito passo a passo o processo de prototipagem das placas que contêm os sensores térmicos, e como ocorre a leitura desses sensores.

- Experimentos com os dispositivos de hardware Xbees Pro, analisando o alcance máximo de comunicação entre eles, além do consumo de energia desses dispositivos móveis, que constituem sua principal restrição.

5.3. Trabalhos Futuros

Como trabalhos futuros têm-se:

- Mais experimentos relacionados ao consumo energético, tentando sempre minimizar o gasto da bateria.

- Testes de interferência serão realizados na freqüência de operação (2.4GHz) dos dispositivos *ZigBee*.

- Integração com outros tipos de sensores, e conseqüentemente a construção de uma nova placa.

- Melhoria das interfaces gráficas da aplicação WEB.

6. Referências

[1] **Datasheet HAT103D1-3990-ADD THERM.**

[2] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin e D. Ganesan, **"Building efficient wireless sensor networks with low-level naming"**, In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles}, Banff, Alberta, Canada, ACM Press, 2001, pp 146-159.

[3] R. Malladi e D. P. Agrawal, **"Current and future applications of mobile and wireless networks"**, Communications of the ACM, ACM Press, ISSN 0001- 0782,vol. 45, no. 10, pp 144-146, 2002. Disponível em:<http://doi.acm.org/10.1145/570907.570947>. Acesso em: abril de 2008.

[4] **IEEE 802.15 WPAN™ Task Group 4 (TG4)**. Disponível em:<http://www.ieee802.org/15/pub/TG4.html>. Acesso em: abril de 2008.

[5] **ZigBee Alliance**. Disponível em:<http://www.zigbee.org/en/index.asp>. Acesso em: abril de 2008.

[6] **UPnP Forum**. Disponível em:<http://www.upnp.org>. Acesso em: abril de 2008.

[7] **Wikipedia**. Disponível em:< http://wikipedia.org/ >. Acesso em: abril de 2008.

[8] **Jini Network Technology.** Disponível em:<http://www.sun.com/software/jini/>. Acesso em: abril de 2008.

[9] A. Williams. **Requirements for Automatic Configuration of IP Hosts**. Technical report, Motorola Inc., 2002.

[10] Microsoft Corporation. **Visual Studio Developer Center**. Disponível em:<http://msdn.microsoft.com/pt-br/vs2005/default(en-us).aspx>. Acesso em: abril de 2008.

[11] Cadsoft. **Eagle Layout Editor**. Disponível em:<http://www.cadsoftusa.com/>. Acesso em: abril de 2008.

[12] IIS.net**. The Official Microsoft IIS Site**. Disponível em:<http://www.iis.net>. Acesso em: abril de 2008.

[13] **Jini**.**org.** Disponível em:< http://www.jini.org/wiki/Main_Page>. Acesso em: março de 2008.

[14] Wikipedia. **Bluetooth**. Disponível em:< http://pt.wikipedia.org/wiki/Bluetooth>. Acesso em: março de 2008.

[15] **Bluetooth Specification**. Disponível em:<https://www.bluetooth.org/spec>. Acesso em: março de 2008.

[16] **JavaServer Pages**. Disponível em:<HTTP://pt.wikipedia.org/wiki/Jsp>. Acesso em: março de 2008.

[17] **IEEE 802.11™ Wireless Local Area Networks**. Disponível em:<http://grouper.ieee.org/groups/802/11/>. Acesso em março de 2008.

[18] **IEEE 802.15 WPAN Task Group 1 (TG1).** Disponível em:<http://www.ieee802.org/15/pub/TG1.html>. Acesso em: março de 2008.

[19] **Estudo e especificação de um sistema de instrumentação para unidades de elevação de petróleo utilizando tecnologia sem fio.** Disponível em:<ftp://ftp.ppgeec.ufrn.br/Mestrado/M182.pdf>. Acesso em: março de 2008.

[20] **Redes de Sensores: ZigBee.** Disponível em:< http://www.ppgia.pucpr.br/~jamhour/Download/pub/Mestrado%202006/ZigBee_Claudia.pdf >. Acesso em: março de 2008.

[21] **Módulos de Comunicação Wireless para Sensores.** Disponível em:< http://paginas.fe.up.pt/~ee02055/RelatorioTEC15.pdf>. Acesso em: março de 2008.

[22] **Redes de sensores sem fio em monitoramento e controle.** Disponível em:<http://www.gta.ufrj.br/ftp/gta/TechReports/Sergio07/Sergio07.pdf>. Acesso em: março de 2008.

[23] **Fusão de Dados Tempo Real em Redes de Sensores Sem Fio Multimídia.** Disponível em:<http://www.das.ufsc.br/~montez/publications/2007%20Webmedia%20Alex.pdf>. Acesso em: março de 2008.

[24] **Monitoramento Ambiental de Salas Limpas através do uso de Redes de Sensores Sem Fio.** Disponível em:<http://www.pad.lsi.usp.br/humanlab/trabalhos/IBERSENSOR_2006_01.pdf>. Acesso em: março de 2008.

[25] **Bluetooth and Sensor Networks: A Reality Check.** Disponível em:<http://www.cens.ucla.edu/sensys03/proceedings/p103-leopold.pdf>. Acesso em: maio de 2008.

[26] **Practical Robust Localization over Large-Scale 802.11 Wireless Networks.** Disponível em:<http://www.mpi-sws.mpg.de/~ahae/papers/mobicom2004.pdf>. Acesso em: maio de 2008.

[27] Wikipedia**. Web Service.** Disponível em:<http://pt.wikipedia.org/wiki/Web_service>. Acesso em: maio de 2008.

[28] **An Infrastructure for Service Oriented Sensor Networks.** Disponível em:<http://www.academypublisher.com/jcp/vol01/no05/jcp01052029.pdf>. Acesso em: maio de 2008.

[29] **Coupling Enterprise Systems with Wireless Sensor Nodes: Analysis, Implementation, Experiences and Guidelines.** Disponível em:<http://www.cobis-online.de/files/publications/n009-f08-decker.pdf>. Acesso em: maio de 2008.

[30] **Tecnologia de Nós Sensores Sem Fio.** Disponível em:<http://homepages.dcc.ufmg.br/~linnyer/ufmgnossensores.pdf>. Acesso em: maio de 2008.

[31] **Monitoramento de Temperatura Utilizando Rede de Sensores Sem Fio em Ambientes Controlados.** Disponível em:<http://www.pad.lsi.usp.br/humanlab/trabalhos/Contecsi_2006_01.pdf>. Acesso em: maio de 2008.

[32] **Tecnologias de Comunicação para Redes Domiciliares.** Disponível em:<http://www.unibratec.com.br/jornadacientifica/diretorio/UFPEPTE.pdf>. Acesso em: maio de 2008.

[33] **A Secure Service Discovery Protocol for MANET.** Disponível em:<http://www.cs.umd.edu/Library/TRs/CS-TR-4498/CS-TR-4498.pdf>. Acesso em: maio de 2008.

[34] **UPnP Networking: Architecture and Security Issues.** Disponível em:<http://www.tml.tkk.fi/Publications/C/25/papers/Haque_final.pdf>. Acesso em: maio de 2008.

[35] **Sharing sensor networks.** Disponível em:<http://eis.comp.lancs.ac.uk/workshops/iwsawc06/papers/20-SensorNetworks.pdf>. Acesso em: maio de 2008.

[36] Y. Gsottberger, et al., **"Embedding Low-Cost Wireless Sensors into Universal Plug and Play Environments,"** EWSN, 2004, pp. 291-306.

[37] **Protocols for service discovery in dynamic and mobile networks.** Disponível em:<http://www.icta.ufl.edu/projects/publications/servicediscovery.pdf>. Acesso em: maio de 2008.

[38] **UPnP™ Device Architecture 1.0.** Disponível em:<http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf>. Acesso em: maio de 2008.

[39] **A Framework for the Integration of Legacy Devices into a Jini Management Federation.** Disponível em:<http://www.aschemann.net/gerd/publications/dsom99.pdf>. Acesso em: maio de 2008.

[40] Beveridge, M. & Koopman, P., **"Jini Meets Embedded Control Networking: a case study in portability failure,"** *Seventh IEEE Workshop on Object-Oriented Real-Time Dependable Systems: WORDS 2002*, San Diego, January 2002.

[41] **Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability.** Disponível em:<http://www.cs.uno.edu/~golden/Stuff/37_richard_g.pdf>. Acesso em: maio de 2008.

[42] Gutierrez, José A. & David B. Durocher (2005), **'On the use of IEEE 802.15.4 to enable wireless sensor network in pulp and paper industry'**, Pulp and Paper Industry Technical Conference, 2005. Conference Record of 2005 Annual pp. 105–110.

[43] Pekhteryev, Georgiy, Zafer Sahinoglu, Philip Orlik & Ghulam Bhatti (2005), **'Image transmission over IEEE 802.15.4 and zigbee networks'**, IEEE International Symposium on Circuits and Systems - ISCAS.

[44] K. Klues, J. Hoffert, O.Orjih, **"Configuring the IEEE 802.15.4 MAC Layer for Single-sink Wireless Sensor Networks"**, Real-Time Systems class project. Professor Chanyang Lu. Washington University, St. Louis, Missouri, December 2005.

[45] LU, G., KRISHNAMACHARI, B., E RAGHAVENDRA, C. S. **Performance Evaluation of the IEEE 802.15.4 MAC for Low-Rate Low-Power Wireless Networks**, 2004.

[46] PETROVA, M., RIIHIJÄRVI, J., MÄHÖNEN, P., E LABELLA, S. **Performance Study of IEEE 802.15.4 Using Measurements and Simulations**. RWTH Aachen University, Germany (2005).

[47] **Arquitetura de Software para Redes de Sensores sem Fios: a Proeminência do Middleware.** Disponível em:<http://www.sbc.org.br/bibliotecadigital/download.php?paper=135>. Acesso em: maio de 2008.

[48] **The WiSe Box : a Multi-performer Wireless Sensor Interface using WiFi and OSC.** Disponível em:<http://hct.ece.ubc.ca/nime/2005/proc/nime2005_266.pdf>. Acesso em: maio de 2008.

[49] ASP.net. **The Official ASP.NET Site**. Disponível em:<http://www.asp.net/>. Acesso em: maio de 2008