UNIVERSIDADE FEDERAL DO PAMPA

BEATRIZ MARTINS DE CARVALHO

DESENVOLVIMENTO DE UM SISTEMA PARA PECUÁRIA DE PRECISÃO CAPAZ DE COLETAR, ARMAZENAR E ANALISAR VARIAÇÕES DAS CONDIÇÕES SUPERFICIAIS DO SOLO EMPREGANDO REDES DE SENSORES SEM FIO.

Bagé
2015

BEATRIZ MARTINS DE CARVALHO

DESENVOLVIMENTO DE UM SISTEMA PARA PECUÁRIA DE PRECISÃO CAPAZ DE COLETAR, ARMAZENAR E ANALISAR VARIAÇÕES DAS CONDIÇÕES SUPERFICIAIS DO SOLO EMPREGANDO REDES DE SENSORES SEM FIO.

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharela em Engenharia de Computação.

Orientador: Dr. Leonardo Bidese de Pinho

Bagé
2015

BEATRIZ MARTINS DE CARVALHO

DESENVOLVIMENTO DE UM SISTEMA PARA PECUÁRIA DE PRECISÃO CAPAZ DE COLETAR, ARMAZENAR E ANALISAR VARIAÇÕES DAS CONDIÇÕES SUPERFICIAIS DO SOLO EMPREGANDO REDES DE SENSORES SEM FIO.

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharela em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 11 de julho de 2015.
Banca examinadora:

_____
Prof. Dr. Leonardo Bidese, de Pinho
Orientador
Engenharia de Computação/Campus Bagé - UNIPAMPA

_____
Prof. MSc. Gerson Leiria Nunes
Engenharia de Computação/Campus Bagé - UNIPAMPA

_____
Prof. Dr. Milton Heinen
Engenharia de Computação/Campus Bagé – UNIPAMPA

Dedico este trabalho em memória da minha vozinha Clotilde Viera Martins, pois em vida sempre me apoiou e incentivou a estudar e hoje sei que ela é uma estrelinha que continua a me guiar no caminho correto.

AGRADECIMENTO

Ao Prof. Dr. Leonardo Bidese de Pinho por toda ajuda, incentivo e principalmente, orientação no desenvolvimento deste TCC.

Aos professores do curso de Engenharia de Computação, por terem compartilhado um pouco do conhecimento comigo, com vocês aprendi muitas coisas e amadureci, hoje sei que me tornei uma Engenheira de Computação. E aos professores da Física, em especial Prof. Wladimir Hernandez Flores e André Gundel por terem auxiliado e dedicado parte do seu tempo e conhecimento nos experimentos realizados neste trabalho.

A todos os colegas de curso, em especial ao Mohamad Badwan que me apadrinhou e é um exemplo de dedicação e superação, por compartilhar seu conhecimento e sempre torcer por mim.

Ao Juliano Borin, "*roommate*" e irmão do coração, pois você esteve comigo em uma fase especialmente delicada da minha vida.

Ao Leandro Gomes e Tassiani Freitas por terem se tornado os meus primeiros amigos quando entrei no curso e estiveram sempre me socorrendo.

Aos meus colegas de pesquisa Odair Santos sem você eu não terminaria o TCC e Moisés Diego Deangelo por compartilhar seu conhecimento e auxiliar com os sensores, ao Érico Nunes meu querido veterano por me ajudar sempre que eu pedisse socorro e ao Jarbas Correia amigo de todas as horas, por tudo.

À minha família, porque sem vocês, eu nada seria. Em especial aos meus pais Cristina Martins de Carvalho e Odilon Gomes de Carvalho, por serem os pais maravilhosos e dedicados que são e por todo apoio e amor que sempre me dão.

À minha maninha Ivi Martins de Carvalho, por ser exemplo de determinação e superação e a todo apoio e amor que tenho recebido,

Ao Rédi dos Santos que apareceu na minha vida e se tornou alguém tão especial, por toda paciência que tem, por ter segurado minha mão em todos os momentos difíceis.

E a todos meus tios e tias que sempre me apoiaram, torceram e me ajudaram quando precisei. Especialmente à Mamãe Eny Carvalho, por toda ajuda e dedicação.

A mente que se abre a uma nova ideia jamais voltará a seu tamanho original.

Oliver Wendell Holmes

RESUMO

As Redes de Sensores Sem Fio (RSSF) possuem várias aplicações que vão desde detecção de incêndio em áreas de difícil acesso até o monitoramento de uma casa inteligente. Particularmente, as RSSF se apresentam como uma solução em potencial para a coleta de dados de temperatura, umidade e luminosidade sobre uma área rural do Bioma Pampa, visando o conforto animal. Neste contexto, este trabalho apresenta o processo de desenvolvimento de um sistema composto por *hardware* e *software* capaz de monitorar uma área do Bioma Pampa utilizando uma Rede de Sensores sem Fio baseada no protocolo ZigBee/802.15.4, mais precisamente os sensores XBee ZB L/T/H (XS-Z16-CB2R), com os quais são desenvolvidos testes preliminares para verificar a viabilidade do uso destes sensores de acordo com os requisitos específicos da aplicação: precisão e frequência de amostragem dos dados mensurados, alcance e durabilidade da bateria. Os testes realizados incluem o de variação na tensão de alimentação, de comparação dos dados medidos com instrumentos calibrados em condições parcialmente controladas, mas sujeitas a efeitos de multicaminho e de diferentes topologias com que estes sensores podem operar para a ampliação da área de cobertura da rede, demonstram preliminarmente o potencial da solução para a coleta de dados em uma área de campo nativo na qual hipoteticamente o efeito do multicaminho será ainda mais reduzido. Por fim são apresentados resultados do uso destes sensores em uma simulação da RSSF em um ambiente externo coletando dados durante um período de tempo.

Palavras-chave: RSSF. Pecuária de precisão. Sensores XBee. Protocolo ZigBee.

ABSTRACT

The Wireless Sensor Networks (WSN) has many applications ranging from fire detection in difficult access areas to the monitoring of smart houses. Particularly, presents itself as a potential solution to the data collection of temperature, humidity and brightness over a rural area of the Pampa biome, aiming animal comfort. In this context, this paper presents the development process of a system composed of hardware and software capable of monitoring an area of the Pampa Biome using an WSN based on ZigBee / 802.15.4 protocol, specifically the sensors XBee ZB L / T / H (XS-Z16-CB2R), with which preliminary tests are developed to verify the feasibility of using these sensors according to specific application requirements: accuracy and sampling frequency of the measured data, range and battery life. Experimental evaluation includes tests (i) to define the input voltage range that supports sensor operation, (ii) to compare data measured from sensor nodes with instruments calibrated in partially controlled conditions but subjected to multipath effects, (iii) to investigate different topologies in which these sensors can operate to expand the coverage area of the network, and (iv) to demonstrate the effectiveness of the client-server application developed to gather data sent from sensor nodes and periodically transmit it to the remote server responsible for the long-term storage. Results suggest the viability of the proposed solution for collecting data over an area of native grassland in which hypothetically the effect of multipath will be further reduced. Moreover, results from the use of such kind of sensors to collect data during a given period of time in an outdoor WSN simulation are presented to confirm the effectiveness of the proposal in an external environment.

Keywords: WSN. Precision livestock. XBee sensors. ZigBee protocol.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

APL - *Application Layer*

APO - *Application Objects*

APS - *Application Support Sublayer*, *Application Sub layer*

AT – *Transparent,* modo transparente

BD - *Baud Rate*

BPSK - *Binary Phase Shift Keying*

CC - Command Sequence Character

Checksum - soma de verificação de erros

cmdData - Identifier-specific Data

cmdID - API Identifier

COPPE - Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

CR - carriage return

DB – *database*

DH - Destination Address High

DL - Destination Address Low

DSSS - *Direct Sequence Spread Spectrum*

Embrapa - Empresa Brasileira de Pesquisa Agropecuária

FAPERGS - Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul

FFD - *Full Function Devices*

Gateway - ponte de ligação

GPIO - *General Purpose Input/Output*

GT - Guard Times

JSON - *JavaScript Object Notation*

LR-WPAN - *Low-Rate Wireless Personal Area Networks*

MAC - *Medium Access Control*

NAD - *Network Address Discovery*

NI - *Node Identifier*

NWK - *Network Layer*

O-QPSK - *Quadrature Phase Shift Keying*

PHY - *Physical*

RD - *Route Discovery*

RF - Radio Frequência

RFD - *Reduced Function Devices*

RO - *Packezation Timeout*

RSSF - Redes de Sensores Sem Fio

SGBD - Sistema Gerenciador de Banco de Dados

UFRJ - Universidade Federal do Rio de Janeiro

Unipampa – Universidade Federal do Pampa

VANT - Veiculo Aéreo Não Tripulado

W3C - *World Wide Web Consortium*

WPAN - Wireless Personal Area Network, Rede pessoal sem fio

WSN - Wireless Sensor Networks

ZDO - *ZigBee Device Object*

SUMARIO

1 INTRODUÇÃO

O Brasil é reconhecido como um dos grandes exportadores de produtos agropecuários no mundo. Parte deste reconhecimento se deve à busca da qualidade dos produtos através do melhoramento genético combinado com o melhoramento das pastagens. Porém, muitas vezes, o incremento de área de pastagens tem implicado na destruição dos biomas. No Rio Grande do Sul, estado onde a economia está fortemente baseada na produção agropecuária, destaca-se o bioma Campos Sulinos (PILLAR et al., 2009). A partir de 2004, os Campos Sulinos foram desmembrados em Bioma Mata Atlântica, formado pelos campos de altitude, e Bioma Pampa, passando a fazer parte da classificação brasileira de Biomas (QUADROS; TRINDADE; BORBA, 2009). O Bioma Pampa, termo indígena que significa região plana, abrange não apenas o estado do Rio Grande do Sul, estendendo-se também pelo Uruguai e a Argentina (DRUMMOND; FRANCO; OLIVEIRA, 2010).

O Bioma Pampa possui paisagens naturais variadas, de serras a planícies, de morros rupestres a coxilhas. As paisagens naturais do Pampa se caracterizam pelo predomínio dos campos nativos, mas há também a presença de matas, formações arbustivas, banhados, afloramentos rochosos, entre outros (MMA, 2014). Neste escopo o uso de pastagens nativas no manejo extensivo tem sido cada vez mais valorizado no mercado, como no caso do gado produzido na Região da Campanha do Rio Grande do Sul. Por outro lado, este bioma deve ser utilizado com cuidado, para evitar a perda da biodiversidade e da qualidade destas pastagens.

A perda de biodiversidade compromete o potencial de desenvolvimento sustentável da região, seja pela perda de espécies de valor forrageiro, alimentar, ornamental e medicinal, seja pelo comprometimento dos serviços ambientais proporcionados pela vegetação campestre, como o controle da erosão do solo e o sequestro de carbono que atenua as mudanças climáticas (MMA, 2014). Assim sendo, especula-se que tais alterações de um bioma poderiam ser observadas por meio do monitoramento de macro e micro variações nas condições naturais superficiais no solo, sendo um dos maiores desafios à coleta periódica de dados a partir de diferentes pontos de amostragem.

Para monitorar estas variações a partir de diferentes pontos de uma determinada área, o emprego de Redes de Sensores Sem Fio (RSSF), também

conhecidas como *Wireless Sensor Networks (WSN)*, se apresenta como uma opção. Esta hipótese se baseia em diversos relatos de experiências bem-sucedidas, cobrindo diferentes aplicações. Conforme Rocha (2014), pode ser citado o uso de RSSF para a detecção de incêndios aonde a distribuição de sensores em uma floresta permite que pontos de calor sejam detectados em pouco tempo e sejam localizados imediatamente e com precisão, viabilizando o controle rápido destes focos antes que se espalhem por uma área muito grande. Outro exemplo também é o monitoramento de movimentos de animais, onde é utilizado para caça e para controle de movimentos de animais perigosos. Pode-se citar também o controle de condições ambientais onde é possível monitorar diversos fatores, tais como o nível de poluição no ar ou na água, a concentração de pesticidas na água e as condições de temperatura e umidade. E, por fim, além destas aplicações, o uso de RSSF no mapeamento da biodiversidade do ambiente onde uma rede de sensores espalhados em determinado ambiente torna possível a detecção de animais e plantas localizados na região.

1.1 MOTIVAÇÃO

Considerando o sucesso em outros contextos, o emprego de RSSF se apresenta como uma solução em potencial para a coleta de dados sobre a superfície do solo, a partir de diferentes pontos de uma determinada área rural pertencente ao bioma em estudo, mais precisamente por meio de sensores capazes de medir temperatura, luminosidade e umidade. Entretanto, para que as RSSF sejam consideradas efetivas para essa aplicação, é necessário um estudo teórico-prático aprofundado sobre detalhes da tecnologia empregada, tais como alcance dos rádios, consumo energético e protocolos de roteamento, tanto em laboratório como a campo (RINGWALD; ROMER, 2007). Como resultado deste trabalho de computação aplicada, espera-se poder colaborar para um melhor entendimento não apenas das condições do solo, mas também permitir avaliar experimentalmente as condições de conforto animal às quais ficam submetidos animais criados sob manejo extensivo, em particular do bioma Pampa.

1.2 OBJETIVOS

Este trabalho possui como objetivo geral desenvolver uma rede de sensores sem fio (RSSF) para a captação contínua de dados sobre a variação existente na umidade, temperatura e luminosidade incidentes no solo de uma área nativa usada para pecuária sob manejo extensivo.

1.2.1 Objetivos específicos

Mais detalhadamente, cabe destacar os seguintes objetivos específicos:

a) Estudar e avaliar experimentalmente o funcionamento de sensores XBee ZB L/T/H (XS-Z16-CB2R) e os protocolos de roteamento disponíveis;
b) Analisar teórica e experimentalmente a viabilidade dos Kits sensores XBee para tratar o problema em questão, em diferentes escalas (dezenas ou centenas de sensores em áreas com dezenas ou centenas de hectares);
c) Analisar, propor e testar formas eficientes de transmitir os dados adquiridos na rede de sensores para a central de armazenamento;
d) Propor estratégias complementares capazes de ampliar a longevidade da rede por meio de protocolos de comunicação de baixo consumo de energia;
e) Investigar outras aplicações de RSSF no contexto de pecuária de precisão;
f) Contribuir para o desenvolvimento regional a partir de pesquisa aplicada em cooperação com especialistas de outras áreas.

1.3 METODOLOGIA

A metodologia adotada para atingir os objetivos elencados parte de um estudo sobre redes, em especial no que tange às topologias e pilhas de protocolos prevalentes em RSSF de forma geral e, em particular, nos nós sensores XBee ZB L/T/H (XS-Z16-CB2R). A partir deste conhecimento de base, são estudados os impactos de fenômenos físicos como a dispersão do sinal, interferências e multicaminhos (reflexão da onda eletromagnética em objetos e no solo) (KUROSE; ROSS, 2006), combinados com uma análise da potência dos equipamentos e antenas disponíveis para avaliação experimental do funcionamento destes sensores.

Adicionalmente, é feita uma análise teórica e experimental para diagnosticar a viabilidade do emprego dos referidos sensores XBee aplicados ao estudo do solo, em diferentes escalas (dezenas ou centenas de sensores em áreas com dezenas ou centenas de hectares).

Após a demonstração da viabilidade de implementação do protótipo da rede de sensores, contemplando a coleta de temperatura, luminosidade e umidade, bem como o agrupamento e transmissão dos dados para um sistema de banco de dados remoto simplificado, é prevista uma instalação da rede a campo (em área da Empresa Brasileira de Pesquisa Agropecuária (Embrapa) Pecuária Sul, preparada por especialistas em pecuária para experimentos em área de pastagem nativa). Este experimento prático incluirá o estudo e avaliação de alternativas para disseminação dos dados obtidos pelos múltiplos sensores. Estes dados serão agrupados e transmitidos para uma central de monitoramento onde serão analisados preferencialmente em tempo real e armazenados em um sistema de banco de dados para futuros estudos de especialistas na área de solos.

1.4 TRABALHOS CORRELATOS

Embora não tenham sido identificados trabalhos específicos que utilizem RSSF para a captação contínua de dados sobre a variação existente na umidade, temperatura e luminosidade em áreas de pastagem, tampouco específicas em áreas de campo nativo do Bioma Pampa, cabe destacar alguns trabalhos encontrados na literatura que apresentam correlação com o presente trabalho.

Souza et al. (2010) apresentou uma proposta de um sistema de automação residencial para iluminação utilizando a tecnologia ZigBee, construindo um protótipo com o objetivo de simular o acionamento de uma lâmpada à distância. A correlação com o presente trabalho está na aplicação da tecnologia para a comunicação remota do controlador com os dispositivos finais.

Rivero et al. (2011) desenvolveu uma RSSF com o objetivo de monitorar equipamentos eletrônicos, permitindo a uma instituição de ensino um controle patrimonial em tempo real. Particularmente, se assemelha ao presente trabalho por utilizar a tecnologia ZigBee como padrão de comunicação entre os dispositivos e a preocupação em minimizar o consumo de energia aumento o tempo de vida de um módulo sensor.

Soares et al. (2012) detalhou o processo de desenvolvimento de um sistema composto por hardware e software capaz de monitorar animais – mais precisamente caprinos - utilizando uma Rede de Sensores sem Fio baseada no protocolo ZigBee 802.15.4, com nós sensores similares ao do presente trabalho (embora não o mesmo kit).

Domingues et al. (2012) detalhou o processo de desenvolvimento de um sistema composto por nós sensores da *Freescale* e software em *Python* que apresenta a localização do nó móvel no campo por meio de “Técnicas de Lateração” com nós sensores da *Freescale*, modelo MC1322x. Posteriormente, Deangelo et al. (2013) apresentaram modificações do sistema para uso com sensores da Digi, modelo XBee Series 2 PRO. Assemelha-se ao presente trabalho por buscarem soluções e apresentarem resultados preliminares que demonstram a viabilidade do emprego de RSSF’s baseadas no Protocolo ZigBee em atividades de agropecuária de precisão, bem como por apresentarem metodologias de desenvolvimento similares.

Segundo Marfievici (2013) apresenta, entre outros dados, o comportamento real de redes de sensores 802.15.4 em ambientes externos, inclusive discutindo o efeito da vegetação na comunicação fornecendo *insights* sobre a confiabilidade e a vida útil da RSSF.

1.5 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em cinco capítulos, conforme segue. O Capitulo 2 – Conceitos Gerais e Revisão da Literatura – consiste numa revisão bibliográfica sobre o contexto do trabalho no que tange aos conceitos básicos necessários para o desenvolvimento da solução integrada de hardware e software proposta. É apresentado o ambiente que o trabalho se propõe a atuar (seção 2.1); as principais características da Rede de Sensores Sem Fio (RSSF) (seção 2.2). A seção 2.3 são apresentados os padrões ZigBee™ e IEEE 802.15.4. Já na seção 2.4 são expostas particularidades dos dispositivos XBee, enquanto que a seção 2.5 detalha o modelo de nós sensor utilizado na pesquisa. Por fim, na seção 2.6 é exposta a rede de sensores proposta no trabalho. No capítulo 3 – Ambiente de Desenvolvimento - são apresentados os equipamentos de medição e ferramentas de *hardware* e *software* utilizados no desenvolvimento do trabalho. Além disso, são

descritos os detalhes de implementação do *software* desenvolvido em *Python* responsável por captar os dados dos sensores XBee presentes na rede e enviá-los para o servidor onde se encontra o sistema gerenciador de banco de dados.

O Capitulo 4 – Resultados Experimentais – caracteriza o ambiente experimental e os resultados dos testes preliminares realizados com os sensores para conhecer a variação de tensão de operação, a precisão dos dados coletados, a área de cobertura e o funcionamento da RSSF em uma simulação em ambiente real.

Por fim, o Capitulo 5 traz as considerações finais sobre o desenvolvimento do trabalho e os trabalhos futuros previstos.

2 CONCEITOS GERAIS E REVISÃO DE LITERATURA

Uma Rede de Sensores Sem Fio (RSSF) ou *Wireless Sensor Network* (WSN) pode ser definida como uma rede de dispositivos (nós), que pode sentir o ambiente e comunicar as informações obtidas a partir do campo monitorado (por exemplo, uma área) através de ligações sem fios. Ao contrário da maioria dos padrões sem fio desenvolvidos como o intuito de oferecer maiores taxas de transferência, o ZigBee procura utilizar baixas taxas de dados. O Padrão ZigBee define as camadas superiores da pilha de protocolos, tendo como base as funcionalidades das camadas inferiores especificadas pelo Padrão IEEE 802.15.4.

Os dispositivos XBee possuem características e modos de operações, que permitem uma estrutura de comunicação (interfaceamento) com outros dispositivos que possuem interface serial (computador, microcontrolador, entre outros).

Serão apresentados os sensores XBee ZB L/T/H (XS-Z16-CB2R), as topologias de redes nas quais é possível utilizar o padrão ZigBee adotado nestes dispositivos e por fim, as plataformas microprocessadas que podem atuar como controlador do coordenador da RSSF.

2.1 REDE DE SENSORES SEM FIO (RSSF)

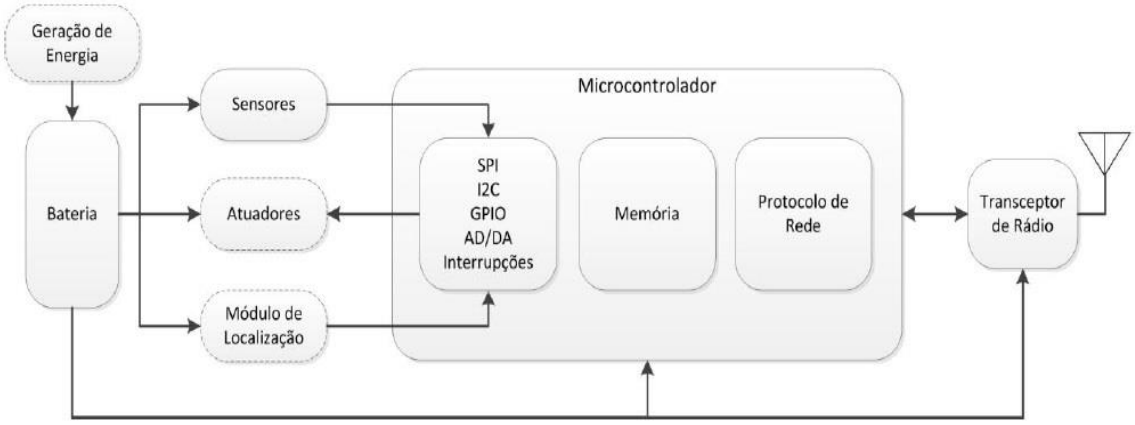
Numa RSSF os dados são encaminhados normalmente através de múltiplos saltos, a um nó “sorvedouro” (algumas vezes denotado como “concentrador”, “controlador” ou “monitor”) que pode estar conectado localmente ou estar conectado a outras redes (por exemplo, à Internet) através de uma ponte de ligação (*gateway)*. Os nós podem ser fixos ou móveis, podendo estar ou não cientes de sua localização, bem como ser homogêneos ou não (BURATTI et al*.* 2009,).

> Um nó sensor é composto por um rádio de comunicação para transmissão e recepção de dados, conectado a uma antena, e um micro controlador para controlar a entrada e saída de dados, efetuar o processamento dos sinais recebidos dos sensores através de conversores analógicos-digitais (RIVEIRO, 2011, p. 22 apud BIGNELL; DONOVAN, 1995).

Alguns sensores podem incluir módulos de localização, geração de energia

e atuadores. Para fins de ilustração, a Figura 1 apresenta uma descrição de alto nível dos componentes de um nó sensor genérico.

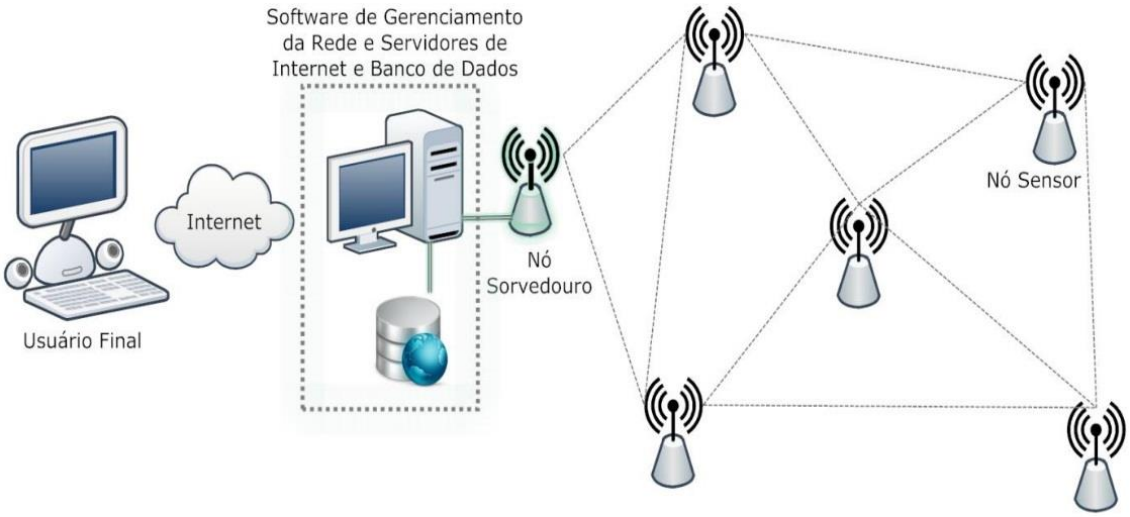
Figura 1 - Descrição de alto nível de um nó sensor genérico



Fonte: Soares (2012, p.18)

Segundo Soares (2012), os dados recebidos pelo sistema são armazenados em banco de dados com o auxílio de um Sistema Gerenciador de Banco de Dados (SGBD). Neste caso, um componente de *software* pode oferecer ao usuário final a funcionalidade de acesso aos dados por meio da Internet, com o auxílio de um servidor web, neste trabalho denominado como “Servidor de Internet”. A Figura 2 apresenta a visão geral da arquitetura e a organização de uma RSSF.

Figura 2 - Visão geral da organização de uma RSSF



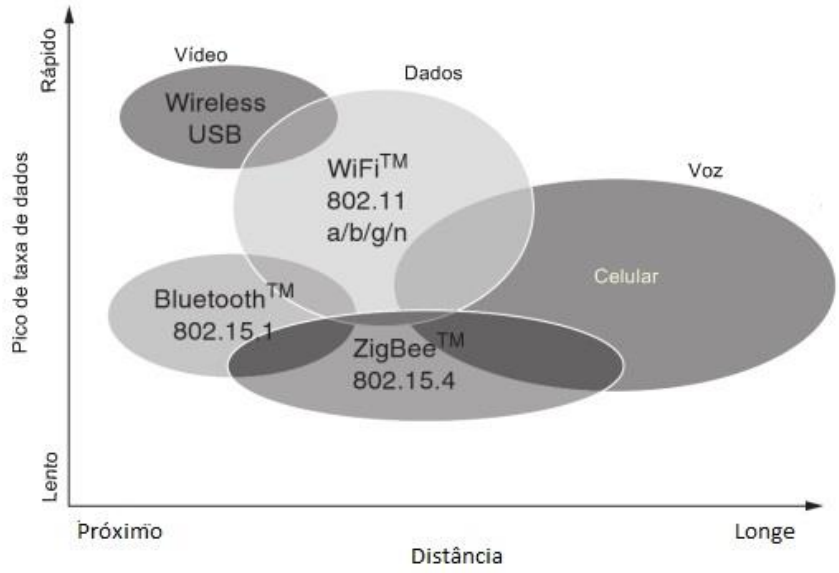
Fonte: Soares (2012, p.18)

2.2 O PADRÃO ZIGBEE™ / IEEE 802.15.4

2.2.1 Surgimento

Com os avanços das tecnologias de redes sem fio, houve o surgimento de várias alternativas e padrões de implementação. A maioria tinha como principal objetivo prover um conjunto de protocolos que garantissem a qualidade para a transmissão de voz ou de dados com altas taxas de transferência, o que tornava os equipamentos caros e com pouco uso para aplicações mais simples (PINHEIRO, 2004). O padrão de rede sem fio ZigBee se encaixa em um mercado que simplesmente não é preenchido por estas tecnologias sem fio, conforme a Figura 3. Enquanto outras tecnologias sem fio focam em fornecer um recurso extra para a Internet ou entregar streaming de mídia de alta definição, o ZigBee foca em controlar uma lâmpada ou enviar dados de temperatura de um termostato. Os outros protocolos sem fio são projetados para funcionar por horas ou talvez dias em baterias, o ZigBee foi projetado para ser executado durante anos (GISLASON, 2008).

Figura 3 - Comparação de tecnologias sem fio



Fonte: Adaptado de Gislason (2008, p.1**)**

Conforme Soares (2012), a tecnologia ZigBee foi concebida para proporcionar uma infraestrutura de rede que apresentasse as seguintes características:

a) formação autônoma de redes com grandes quantidades de dispositivos em

uma área de cobertura, tal que os dispositivos pudessem se comunicar de forma confiável;

b) baixo consumo de energia associado a um reduzido custo de infraestrutura, complexidade e tamanho;
c) taxa de transmissão de dados relativamente baixa;
d) protocolo padronizado e aberto que permitisse a comunicação entre produtos de diferentes fabricantes para um mercado global.

Com esse intuito surgiu a *ZigBee Alliance* (www.zigbee.org), uma associação de empresas que trabalham em conjunto para desenvolver padrões (e produtos) para redes de baixa potência sem fio confiável e econômica (Baronti et al, 2007).

A tecnologia ZigBee provavelmente será incorporada em uma ampla gama de produtos e aplicações em mercados de consumo, comerciais, industriais e governamentais em todo o mundo. As principais áreas de atuação desta tecnologia são apresentadas na Figura 4.

Figura 4 - Área de atuação do ZigBee

Fonte: Soares (2012, p.21)

2.2.2 Organização e arquitetura do Padrão ZigBee/IEEE 802.15.4

O Padrão IEEE 802.15.4 define as características da camada física, conhecida como *Physical* (PHY), e da camada de controle de acesso ao meio, conhecida como *Medium Access Control* (MAC), para redes sem fio pessoais com baixa taxa de transmissão denominadas *Low-Rate Wireless Personal Area Networks*

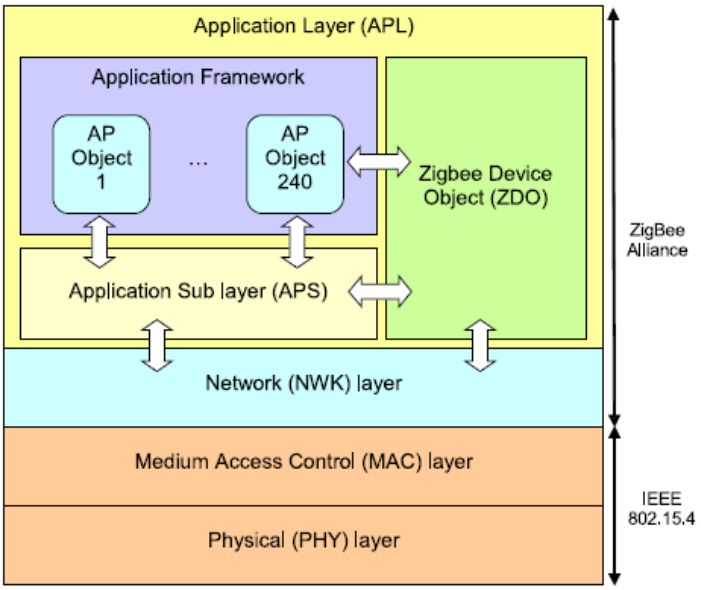
(LR-WPAN).

O Padrão ZigBee define as camadas superiores da pilha de protocolos. A *Network Layer (*NWK) é a camada de rede responsável pela organização e prestação de serviço roteamento em uma camada de múltiplos saltos, tendo como base as funcionalidades das camadas inferiores especificadas pelo Padrão IEEE 802.15.4. A *Application Layer* (APL) é a camada de aplicação que fornece um *framework* (conjunto de definições) para o desenvolvimento de aplicações distribuídas e a comunicação. Segundo Baronti et al (2007) a APL compreende:

a) *Application Framework* responsável pela camada de Quadros de Aplicação; o
b) *ZigBee Device Objects* (ZDO), responsável pela camada de Objetos de Dispositivos ZigBee e o
c) *Application Sub Layer* (APS), responsável pela subcamada de aplicação.

Para facilitar o entendimento, é utilizada neste trabalho o termo ZigBee para definir todo o conjunto de camadas, incluindo as definidas pelo IEEE 802.15.4. A Figura 5 apresenta uma visão geral da pilha de protocolos utilizada no Padrão ZigBee.

Figura 5 - Diagrama da arquitetura e pilha de protocolo ZigBee



Fonte: Baronti et al. (2007, p. 1659).

O ZigBee não se enquadrada exatamente no modelo OSI de sete camadas de rede, mas detêm alguns elementos, incluindo as camadas PHY, MAC e NWK. As Camadas 4-7 (transporte, sessão, apresentação e aplicação) são envolvidas nas camadas APS e ZDO no modelo ZigBee (Gislason 2008).

A camada física (PHY) suporta funcionalidades para seleção de canal, estimativas de qualidade do link, detecção de nível de energia e avaliação de canais disponíveis. Esta camada atua em três faixas de frequência:

a) uma frequência de 2450 MHz (com 16 canais de 250 Kbps);
b) uma frequência de 915 MHz (com 10 canais de 40 Kbps) e
c) uma frequência de 868 MHz (com 1 canal 20 Kbps), todos usando o modo de acesso *Direct Sequence Spread Spectrum* (DSSS) responsável pelo espalhamento espectral por sequencia direta.

A modulação do sinal adotada para a banda de 2450 MHz emprega o *Offset Quadrature Phase Shift Keying* (O-QPSK), enquanto as bandas de 868 e 915 MHz empregam o *Binary Phase Shift Keying* (BPSK) (Baronti, 2007).

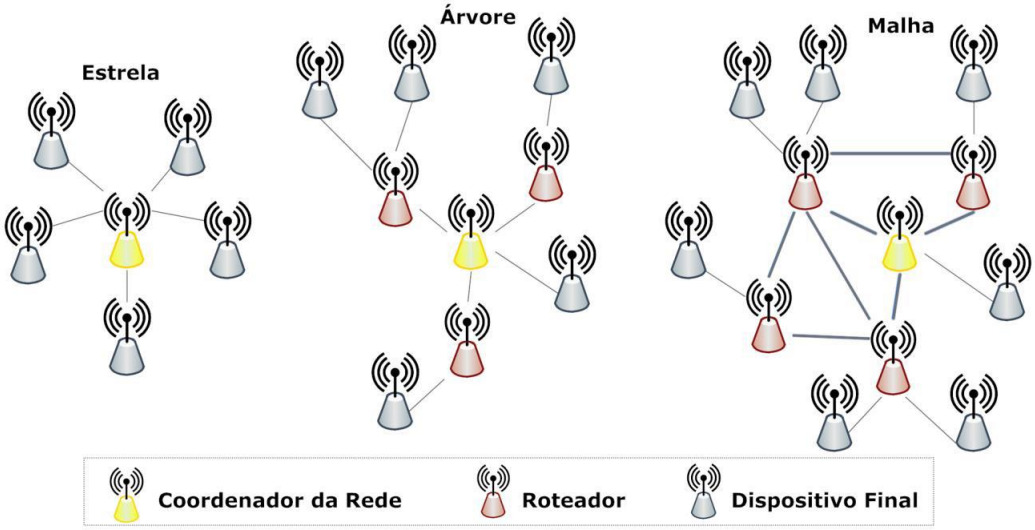
A camada MAC define dois tipos de nós:

a) *Full Function Devices* (FFDs), responsáveis pelos dispositivos que estão equipados com as funções completas da camada MAC, podendo assim agir como coordenador ou roteador da rede e os
b) *Reduced Function Devices* (RFDs), responsável pelos dispositivos de função reduzida que agem apenas como dispositivos finais.

Os FFDs enviam *frames* de reconhecimento (*beacons*) que provê a sincronização, comunicação e serviços da rede e um coordenador PAN é selecionado para administrar o funcionamento da rede.

A camada de rede (NWK) identifica três tipos de dispositivos. Um dispositivo final corresponde a um RFD ou FFD atuando como um dispositivo final. Um roteador correspondente a um FFD com capacidades de roteamento. E um coordenador (somente um na rede) corresponde a um FFD gerenciando toda a rede. Além da topologia star (estrela), que mapeia naturalmente para a topologia correspondente em IEEE 802.15.4, a camada de rede também suporta topologias mais complexas como a *tree* (árvore) e a *mesh* (malha). A Figura 6 apresenta exemplos destas topologias.

Figura 6 - Topologias da rede ZigBee



Fonte: Soares (2012, p.24)

Entre as funcionalidades fornecidas pela camada estão: múltiplos saltos de roteamento, descoberta de rota e manutenção, segurança e união/saída (*joining/leaving*) de uma rede, com atribuição de endereços (16 bits) para os dispositivos recém-unidos (Baronti et al, 2007).

A camada de aplicação (APL) é composta pelos *Application Objects* (APOs) distribuídos em vários nós na rede. Um APO é um pedaço de software (a partir de um desenvolvedor de aplicativos) que controla uma unidade de hardware (transdutor, interruptor, lâmpada) disponível no dispositivo. O *ZigBee Device Object* (ZDO) é um objeto especial que proporciona serviços aos APOs permitindo-lhes descobrir os dispositivos da rede e os serviços que eles implementam. O ZDO também oferece serviços de comunicação, de rede e de gerenciamento de segurança. A *Application Support Sublayer*, também chamada de *Application Sub layer* (APS) presta serviços de transferência de dados para o APOs e o ZDO conforme a Figura 5 onde são ilustrados os vários componentes na camada de aplicação.

Uma aplicação ZigBee deve estar de acordo com um perfil de aplicação existente, aceito pela ZigBee Alliance. Este perfil define os formatos de mensagens e protocolos para interações entre APOs que formam coletivamente uma aplicação distribuída. O perfil do *Application Framework* permite a diferentes desenvolvedores construir e vender de forma independente os dispositivos ZigBee que poderão interoperar uns com os outros em um determinado perfil de aplicação.

Um perfil de aplicação especial, *Device Profile* (perfil do dispositivo), deve

ser implementado por todos os nós em uma rede ZigBee. O perfil do dispositivo requer de seus objetos de execução (ZDOs) dispositivos para suporte/procedimentos de detecção, por meio do qual um nó tenta descobrir os nós existentes na rede - nós finais ativos e/ou os serviços que implementam.

Procedimentos de detecção são cruciais para o endereçamento do APO. No modo de endereçamento direto, uma mensagem é endereçada para um endereço específico de destino (endereço de rede de 16 bits). Neste caso, tanto o número do nó final e o nó de envio serão descobertos através dos serviços de detecção do ZDO. Por outro lado, no modo de endereçamento indireto, é requerido apenas que o remetente forneça um ID de cluster, cabendo ao roteador (*router*) ZigBee (ou um coordenador) auxiliar na localização do(s) nó(s) de destino da mensagem (isto é possível graças à APS do roteador ZigBee que mantém uma tabela de ligação (*binding table*) associando a um objeto de dados (tuplas) - endereço de origem, *endpoint* de origem, ID de cluster - as listas de tuplas - endereço de destino, *endpoint* de destino), uma para cada dispositivo que a mensagem deve alcançar. *Endpoints* são pontos finais destino atribuído para uma única aplicação, para a entrega de mensagens. O *handler* é livre para atribuir manualmente um significado para o ponto final de origem em pacotes recebidos, mas não é o único (Digi). A adição e remoção de entradas na tabela de ligação são comandadas pelo ZDO em resposta às solicitações locais / remotas, conforme definido no perfil do dispositivo (Baronti et al, 2007).

2.3 DISPOSITIVOS XBEE

Dentre os vários dispositivos de *hardware* baseados no protocolo ZigBee, um modelo bastante conhecido é o XBee, atualmente fabricado pela líder de mercado Digi International (www.digi.com). Os módulos XBee são compostos, basicamente, por um microcontrolador e um transceptor. Segundo Soares (2012) o microcontrolador contém o *firmware* com a implementação do protocolo ZigBee e a especificação do comportamento do dispositivo cujas características são definidas por Boechat (2010) como:

a) Coordenador (*coordinator*) - Responsável por selecionar o canal e PAN ID, ele inicia uma nova PAN e permite que roteadores e dispositivos finais se juntem à PAN. O coordenador transmite e recebe transmissões de dados de

Rádio Frequência (RF) e auxilia no encaminhamento de dados através da topologia de rede escolhida. Devido as suas funcionalidades é ideal que o coordenador tenha uma alimentação que não seja por bateria;

b) Roteador (*router*) - Deve participar de uma PAN ZigBee antes que possa operar. Ao se juntar à PAN, o roteador permite que outros roteadores e dispositivos finais se unam à PAN por meio deste e não do próprio coordenador. O roteador também transmite e recebe transmissões de dados em RF e auxilia no encaminhamento de dados através da topologia de rede escolhida. Devido as suas funcionalidades serem parecidas com o coordenador, não é aconselhável entrar em modo de *sleep (*baixo consumo), o que implica na necessidade de uma fonte de energia mais duradoura;
c) Dispositivo Final (e*nd device*) - Deve participar de uma PAN ZigBee para poder funcionar, de forma análoga a um roteador. No entanto, o *end device* não permite a adesão de outros dispositivos à PAN e não pode auxiliar no encaminhamento de dados através da rede. Um dispositivo final transmite e recebe transmissões de dados em RF mesmo alimentados por baterias comuns, por longos períodos de tempo graças ao seu modo *sleep* estado de baixo consumo de energia quando não estiver em uso.

Cabe ao dispositivo (coordenador ou roteador) que permitiu que o dispositivo final aderisse à rede gerenciar todos os dados endereçados a este nó enquanto este está no modo de baixo consumo. Em função disso, este dispositivo é conhecido como pai do dispositivo final (Boechat, 2010).

2.3.1 Endereçamento físico

O Protocolo 802.15.4 sob o qual Protocolo ZigBee é construído especifica dois tipos de endereço: o MY e o Número Serial.

MY (16 bits) é o endereço de rede único, e é distribuído automaticamente pelo coordenador assim que o nó entra na PAN. Existem duas condições para atribuir a um nó um novo endereço de rede:

a) se um dispositivo final não pode se comunicar com seu coordenador, ele precisa sair da rede e retornar para encontrar um novo coordenador;
b) se houve alterações do tipo roteador para dispositivo final ou vice-versa, o dispositivo deixará a rede e voltará como um dispositivo novo.

O protocolo define que os dados sejam enviados para um destino através do MY. Para que isso ocorra, todos os dispositivos devem ser detectados antes de transmitir os dados.

Número Serial (64 bits), também conhecido como Endereço de Dispositivo, é único e invariável para cada dispositivo fabricado.

2.3.2 Tipo de transmissão

São permitidos dois tipos de transmissão: broadcast e unicast.

Transmissões *broadcast* – devem ser propagadas em toda a rede de modo que todos os nós recebam a transmissão. Para que isso ocorra, todos os dispositivos (coordenador e roteadores) que recebam a transmissão irão retransmitir o pacote *broadcast* três vezes. Cada nó que transmite um pacote irá criar uma entrada em uma tabela de transmissão *broadcast*. Esta tabela mantém oito entradas persistentes por oito segundos. Para cada transmissão, a pilha ZigBee deve reservar espaço na memória para uma cópia do pacote pois esta cópia é usada para retransmitir o pacote caso necessário. Na transmissão broadcast utiliza-se o endereço de 64 bits de destino igual a 0x000000000000FFFF.

Transmissões *unicast* – são sempre dirigidas ao endereço MY do dispositivo. Como este endereço não é estático, os dispositivos ZigBee empregam o *Network Address Discovery* (NAD), responsável pela descoberta de endereços de rede, identificando assim o MY que corresponde a um Número Serial, bem como, através do *Route Discovery* (RD), responsável pela descoberta de rota, estabelecem uma rota até o dispositivo (Boechat, 2010).

2.3.3 Modos de operação do XBee

Os módulos XBee suportam duas interfaces de comunicação: *Transparent* (AT) e *Application Programming Interface* (API).

- Modo de operação transparente (AT)

Quando se utiliza o modo de operação transparente, os módulos atuam como um substituto da linha de comunicação serial. Todos os dados recebidos pelo

UART no pino de entrada (DIN) são transmitidos na fila de RF para o módulo de destino. Quando os dados de RF são recebidos através da antena, são enviados ao pino de saída (DOUT). Os parâmetros de configuração do módulo são configurados usando o comando interface de modo AT (DIGI, 2014).

Os dados estão sendo armazenado no buffer de recebimento da serial até que uma das seguintes situações faz com que os dados sejam empacotados e transmitidos. Nenhum *byte* da serial será recebido por uma determinada quantidade de tempo determinado pelo parâmetro RO (*Packezation Timeout*), se o RO=0, o empacotamento começa quando o *byte* é recebido. Quando o modo de sequência de comando (*Guard Times* (GT) + *Command Sequence Character* (CC) + GT) é recebido, qualquer byte armazenado no buffer da porta de entrada da serial, antes da sequência é transmitido.

Módulos de RF que contêm as seguintes versões de *firmware* utilizam o modo transparente: 20xx (AT *coordinator*), 22xx (AT *router*) e 28xx (AT *end device*).

- Modo de operação API

Este modo é uma alternativa ao funcionamento no modo transparente. A API é baseada em quadros (*frames*) e estende o nível a que um aplicativo *host* pode interagir com os recursos de rede do módulo. Quando em modo API, todos os dados que entram e saem do módulo estão contidos em quadros que definem as operações ou eventos dentro do módulo (DIGI, 2014).

A transmissão de *frames* de dados que são recebidos através do pino DIN (pino 3) incluem:

a) transmissão de *frames* de dados através de RF;
b) *frame* de comando (equivalente a comandos AT).

O recebimento de *frames* de dados que são enviados através do pino DOUT (pino 2) incluem:

a) Recebimento de *frames* de dados através de RF;
b) resposta de comando;
c) notificações de eventos, tais como reinicialização, associação, dissociação, etc.

A opção de operação API facilita muitas operações, tais como os exemplos citados abaixo:

a) transmissão de dados para vários destinos, sem entrar no Modo de Comando;
b) receba status de sucesso / fracasso de cada pacote de RF transmitido;
c) identificar o endereço de origem de cada pacote recebido.

Módulos de RF que contêm as seguintes versões de *firmware* oferecem suporte à operação em modo API: 21xx (*coordinator* API), 23xx (*router* API) e 29xx (*end device* API).

2.3.4 Configuração dos módulos XBee

A Digi produz duas famílias de sensores XBee: XBee *Series* 1 e XBee *Series* 2. Embora similares, trazem diferenças significativas no que se refere ao alcance de transmissão e ao consumo energético. A Tabela 1 sintetiza estas diferenças, as quais justificam a escolha do XBee Series 2 para o desenvolvimento deste trabalho.

Tabela 1 - Tabela de comparação das diferenças dos XBee Series

	XBee Series 1	**XBee Series 2**
Indoor/Urban range	até 100 ft. (30m)	até 133 ft. (40m)
Outdoor RF line-of-sight range	até 300 ft. (100m)	até 400 ft. (120m)
Transmit Power Output	1 mW (0dbm)	2 mW (+3dbm)
RF Data Rate	250 Kbps	250 Kbps
Receiver Sensitivity	-92dbm (1% PER)	-98dbm (1% PER)
Supply Voltage	2.8 - 3.4 V	2.8 - 3.6 V
Transmit Current (typical)	45 mA (@ 3.3 V)	40 mA (@ 3.3 V)
Idle/Receive Current (typical)	50 mA (@ 3.3 V)	40 mA (@ 3.3 V)
Power-down Current	10 uA	1 uA
Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.0960" x 1.087"	0.0960" x 1.087"
Operating Temperature	-40 para 85 C	-40 para 85 C
Antenna Options	PCB, Integrated Whip, U.FL, RPSMA	PCB, Integrated Whip, U.FL, RPSMA
Network Topologies	Point to point, Star, Mesh (with Digi Mesh firmware)	Point to point, Star, Mesh
Number of Channels	16 Direct Sequence Channels	16 Direct Sequence Channels
Filtration Options	PAN ID, Channel & Source/Destination	PAN ID, Channel & Source/Destination

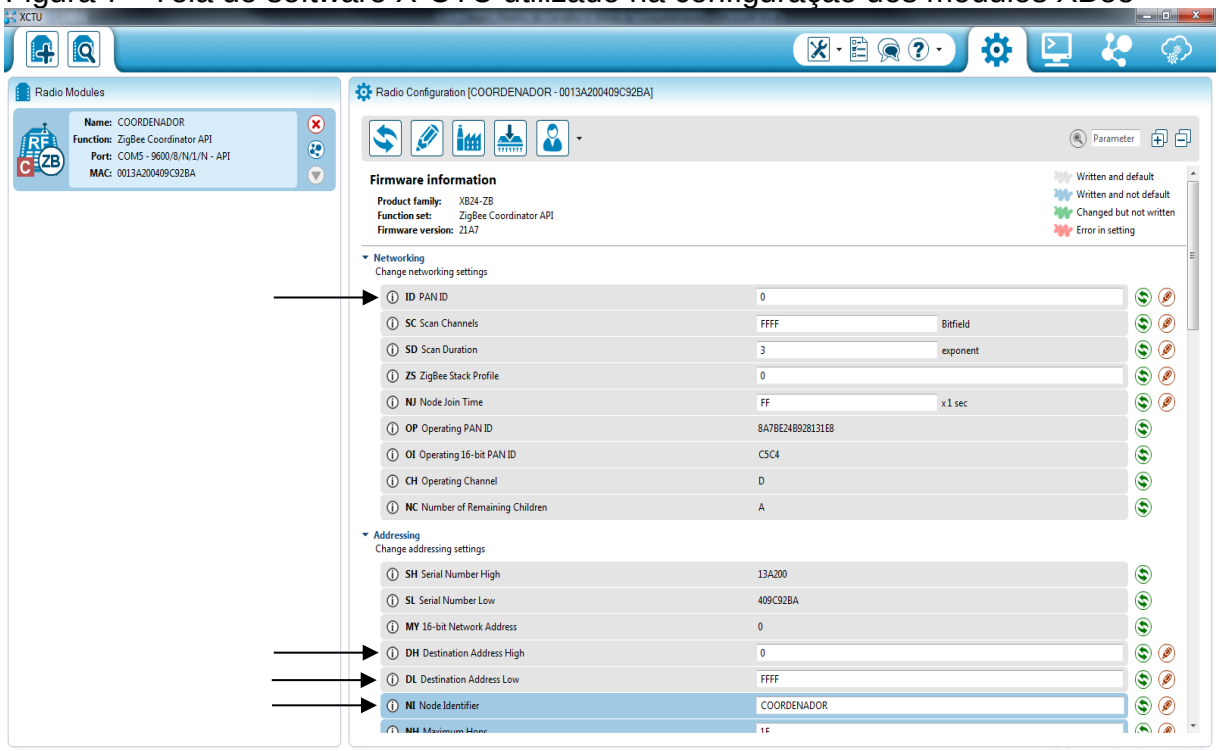
Fonte: Digi

A Digi disponibiliza, através da sua página na Internet, a ferramenta X-CTU (Figura 7), atualmente na versão 6.2.0. Por meio da interface intuitiva desta

ferramenta, o usuário, antes de criar a PAN, deve carregar previamente para os dispositivos um *firmware* que contém as informações sobre o comportamento esperado para o nó na rede, ou seja, determinar se ele é um Coordenador, Roteador ou Dispositivo Final, bem como configurar todos os parâmetros do dispositivo.

As setas incluídas na Figura 7 destacam alguns parâmetros importantes do módulo XBee, à medida que o usuário solicita a leitura ou alteração de parâmetros, o software envia automaticamente os respectivos comandos AT para XBee pela interface RS232 e exibe os resultados na tela. Dentre os parâmetros configuráveis, destacam-se: nome do nó, endereço de destino, velocidade de comunicação RS232 (*baud rate*), comportamento dos pinos de entrada e saída, chave secreta de criptografia, ID da rede, entre outros.

Figura 7 - Tela do software X-CTU utilizado na configuração dos módulos XBee



Fonte: Própria autora (2014).

- Comandos AT no modo de operação transparente

Para modificar ou ler os parâmetros do módulo de RF, o módulo deve primeiramente entrar em *Command Mode* (Modo de Comando), um estado em que os caracteres da serial recebidos são interpretados como comandos (DIGI, 2014). Para tanto, é necessário enviar uma sequência de comando composta por três

caracteres "+ + +" e observar o *guard times* e depois os caracteres de comando.

O Padrão da sequência do modo de comando é:

a) nenhum caractere enviado por um segundo [GT (*Guard Times*) parâmetro = 0x3E8];
b) entrada de três caracteres de mais ("+++") dentro de um segundo [CC (*Command Sequence Character*) parâmetro = 0x2B.];
c) nenhum caractere enviado por um segundo [GT (*Guard Times*) parâmetro = 0x3E8].

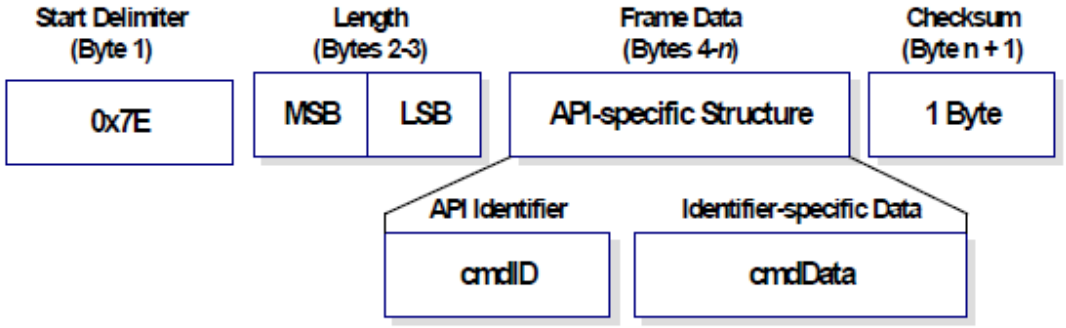
Uma vez que a sequência do modo de comando AT tenha sido emitida, o módulo envia um "OK\r" no pino DOUT. O envio dos caracteres "OK\r" pode ser adiado se o módulo não terminou a transmissão de dados seriais recebidos. Quando o modo de comando foi inserido, o modo de comando temporizador é iniciado (comando CT) e o módulo é capaz de receber comandos AT no pino DIN. Todos os valores dos parâmetros na sequência podem ser modificados de modo a refletir as preferências do utilizador. Cabe destacar que, em caso de falha no envio dos comandos, é importante verificar a taxa de transmissão já que é comum ocorrer incompatibilidade pois padrão o BD (*Baud Rate*) tem como parâmetro = 3 (9600 bps).

- Comandos AT no modo de operação API

O modo de operação API exige que as comunicações com um módulo XBee sejam empacotadas em uma interface estruturada. A API especifica que os comandos, respostas dos comandos e mensagens de status do módulo são enviados e recebidos a partir do módulo usando um *UART Data Frame* (quadro de dados UART). Com este modo de operação é permitido que comandos AT sejam executados tanto localmente quanto remotamente, o que amplia o nível de gerenciamento e a funcionalidade da rede (Digi, 2014).

A estrutura geral de um pacote API é apresentada na Figura 8. Pode-se perceber que o primeiro byte é o delimitador de início, que possui o valor 7E em hexadecimal. Qualquer dado recebido antes do delimitador é descartado. Os bytes 2 e 3 informam o tamanho do pacote que está sendo recebido. Os bytes 4 a n compõem o pacote de dados e definem a operação a ser realizada.

Figura 8 - *Frame* de dado UART e estrutura específica API



Fonte: Digi (2014, p.100)

O último byte contém o *checksum* (soma de verificação de erros) do pacote de dados, calculado pela Equação 1. Caso o recebimento dos dados ou o valor do *checksum* esteja incorreto, um pacote de status indicando a natureza do erro é retornado.

$$\boldsymbol{Checksum} = \boldsymbol{0xFF} - \sum_{i=4}^{n} \boldsymbol{byte}[i] \qquad \text{Equação 1}$$

Os dados mais importantes do pacote API estão contidos nos bytes 4 a n. O byte 4 responsável pelo frame *cmdID* (*API Identifier*) informa o tipo de mensagem da API que será contida nos bytes 5 a n responsável pelo frame *cmdData* (*Identifier-specific Data*). É importante ressaltar que os valores são enviados em *big endian* (o byte menos significativo é enviado primeiramente). Os módulos XBee suportam os frames API apresentados na Figura 9. O formato do quadro de dados, bytes 5 a n (cmdData), varia de acordo com o tipo de operação a ser realizada.

Figura 9 - Nome do API e valores correspondentes

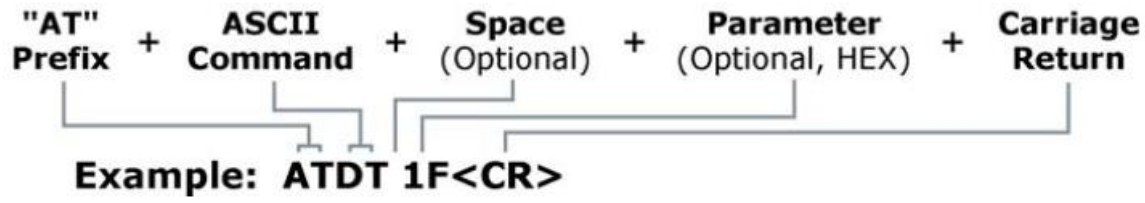
API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1
Many-to-One Route Request Indicator	0xA3

Fonte: Digi (2014, p.100).

- Comandos AT Locais

A Figura 10 apresenta a estrutura de um comando AT aonde cada comando deve possuir o prefixo “AT” concatenado com o nome do comando a ser executado (dois caracteres ASCII). Em seguida, se necessário, é colocado o parâmetro do comando. Por fim, é confirmada a execução do comando pelo envio de um byte de retorno de carro (*carriage return* <CR>).

Figura 10 - Sintaxe para envio do comando AT



Fonte: Digi (2014, p.32).

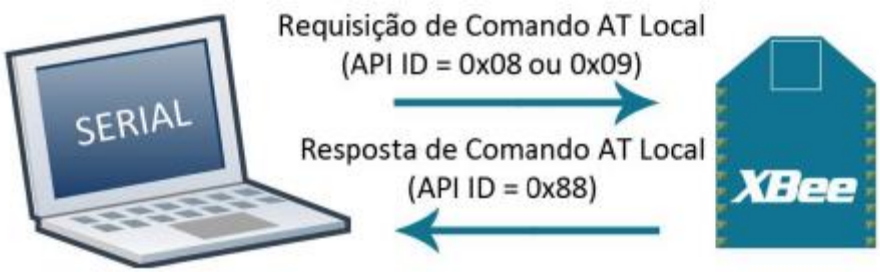
Quando um comando é executado sem a passagem de parâmetro, o módulo XBee interpreta que o usuário deseja consultar o valor daquele comando e retorna uma mensagem com o respectivo resultado. Caso algum parâmetro seja fornecido, o dispositivo altera o valor do referido parâmetro e retorna uma confirmação (“OK”). Se o parâmetro ou o comando possuir um valor inválido, uma mensagem de erro (“ERROR”) é retornada.

É importante destacar que os comandos AT, utilizando o modo de operação Transparente (AT), só serão executados localmente, ou seja, diretamente por meio da interface RS232 do dispositivo. Esta característica limita a capacidade de gerenciamento da rede, uma vez que o usuário não poderá configurar as funcionalidades ou utilizar os periféricos do dispositivo à distância (Soares, 2012).

Ao executar a opção local de comandos AT, é considerado que o XBee está conectado diretamente, através de sua interface serial, ao dispositivo que deseja configurá-lo. Esse dispositivo pode ser um computador, micro controlador ou qualquer outro que possua interface de comunicação serial.

A Figura 11 apresenta a troca de um *frame* API que ocorre no UART ao enviar um pedido de comando AT para ler ou definir um parâmetro de módulo. A resposta pode ser desativada, definindo o *frame* ID para 0 no pedido (Digi,2014).

Figura 11 - Troca de pacotes API para a execução local de comando AT



Fonte: Soares (2012, p.33)

O pacote de requisição para a um comando AT local é descrito na Tabela 2, onde é requisitada, através do comando NI (*Node Identifier*), a identificação do sensor.

Tabela 2 - Estrutura de um pacote API de requisição para comando AT local

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote		MSB 1	0x00	Quantidade de *bytes* entre o tamanho e o *checksum.*
			LSB 2	0x04	
	Dado especifico do pacote	Tipo do pacote API	3	0x08	Pacote do tipo 0x08 (comando AT local)
		ID do quadro	4	0x01	O usuário pode identificar o pacote com um número arbitrário. Se for configurado como 0, nenhuma resposta de *status* é enviada.
		Comando AT	5	0x4E	Nome do Comando AT – Dois caracteres ASCII. Neste exemplo executa-se o comando NI – *Node Identifier*
			6	0x49	
		Valor do Parâmetro (Opcional)			Quando presente indica que se deseja alterar o parâmetro do módulo para o valor fornecido.
	Checksum		7	0x5F	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*.

Fonte: Digi (2014, p. 104)

O pacote de resposta a um comando AT local é muito semelhante ao pacote descrito na Tabela 1. A única diferença é a adição de um byte de status que informa se foi possível realizar a operação. A Tabela 3 apresenta a estrutura de um pacote de resposta (0x88). Cabe observar, neste exemplo, a presença do nome do dispositivo (“SENSOR”), retornado como parâmetro do comando NI (*Node Identifier*) e o byte de status que informa o sucesso na operação.

Tabela 3 - Estrutura de um pacote API de resposta para um comando AT local

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de bytes entre o tamanho e o *checksum*. Neste exemplo 11 bytes
		LSB	2	0x0B	
	Dado especifico do pacote	Tipo do pacote API	3	0x88	Pacote do tipo 0x88 (comando AT local)
		ID do quadro	4	0x01 (R)	O usuário pode identificar o pacote com um número arbitrário. Se for configurado como 0, nenhuma resposta de status é enviada.
		Comando AT	5	0x4E (N)	Nome do Comando AT – Dois caracteres ASCII.
			6	0x49 (I)	
		Status do Comando	7	0x00	0 = OK 1 = ERRO 2 = Comando Inválido 3 = Parâmetro Inválido 4 = Falha na Transmissão
		Valor do Parâmetro	8	0x53	Se um parâmetro for configurado, este campo não retorna nada. Se for realizada uma consulta, o valor do parâmetro é retornado.
			9	0x45	
			10	0x4E	
			11	0x53	
			12	0X4F	
			13	0x52	
	Checksum		14	0x05	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*

Fonte: Digi (2014, p.111).

- Comandos AT Remotos

A principal vantagem proporcionada pelo uso do modo de operação API, em relação ao modo Transparente (AT), é a capacidade de execução remota (à distância) de comandos AT. Isto inclui, dentre outras funcionalidades, a configuração de parâmetros de rede, acionamento de saídas digitais, leitura de entradas analógicas (sensores) e recebimento de mensagens de status da execução destes comandos.

A Figura 12 apresenta uma execução remota de comandos AT, na qual um dispositivo que possui interface serial (computador, microcontrolador, etc.) é responsável por montar o pacote de requisição de Comando AT Remoto (0x17) e

enviá-lo através da interface serial para um módulo XBee. Este módulo verifica o endereço de destino e encaminha a mensagem via RF para o destinatário que, por sua vez, responde com um pacote do tipo 0x97 informando o resultado da operação Soares (2012).

Figura 12 - Troca de pacotes API para execução remota de comandos AT

Fonte: Soares (2012, p.35)

A estrutura de um pacote de requisição de comando AT remoto é apresentada na Tabela 4 e, como exemplo, configura-se o pino “D2” como entrada analógica. Conforme pode ser observado, na requisição de comando AT remoto aparecem novos campos, como os endereços do destinatário e as opções de comando remoto que permitem, dentre outras funcionalidades, o enfileiramento da requisição.

Tabela 4 - Estrutura de um pacote API de requisição para um comando AT Remoto

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de bytes entre o tamanho e o *checksum*. Neste exemplo 16 bytes
		LSB	2	0x10	
	Dado específico do pacote	Tipo do pacote API	3	0x17	Pacote do tipo 0x17 (comando AT local)
		ID do quadro	4	0x02	Se for configurado como 0, nenhuma resposta de status é enviada.
		Endereço de 64 bits de destino MSB (5 - 11) LSB (12)	5	0x00	Endereço de 64 bits do destinatário. Os seguintes endereços também são suportados: 0x0000000000000000: Reservado para o coordenador da rede 0x000000000000FFFF: Endereço de Broadcast
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x40	
			11	0x11	
			12	0x22	
		Endereço de 16 bits de destino (MY) MSB (13) LSB (14)	13	0xFF	Endereço de 16 bits do destinatário. Deve ser igual a 0xFFFE se o endereço do destinatário é desconhecido ou a transmissão broadcast.
			14	0xFE	
		Opções de Comando Remoto	15	0x02	Configurado bit a bit, este campo pode habilitar algumas funcionalidades: 0x01–Desabilita resposta ACK 0x02 –Executa o comando imediatamente (Apply Changes). Se este bit não for setado, a requisição fica enfileirada e comando AC deve ser enviado para confirmar a execução. 0x40 - Usa o tempo limite de transmissão estendida para este destino.
		Comando AT	16	0x44	Nome do Comando AT – Dois caracteres ASCII.
			17	0x32	
		Valor do Parâmetro		0x02	O comando D2, admite os seguintes argumentos: 0x00 – Desabilitado 0x01 – Não utilizado 0x02 – Conversor Analógico-Digital 0x03 – Entrada Digital 0x04– Saída Digital em Nível Baixo 0x05–Saída Digital em Nível Alto
	Checksum		18	0x39	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao checksum.

Fonte: Adaptado de Digi (2014, p. 109).

O pacote de resposta a um comando AT remoto é semelhante ao pacote de

requisição apresentado anteriormente. A única diferença é a substituição do campo de opções do comando remoto pelo campo de status do comando. A Tabela 5 apresenta esta estrutura.

Tabela 5 - Estrutura de um pacote API de resposta a um comando AT Remoto

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de *bytes* entre o tamanho e o *checksum.* Neste exemplo 16 bytes
		LSB	2	0x0F	
	Dado específico do pacote	Tipo do pacote API	3	0x97	Pacote do tipo 0x97 (comando AT local)
		ID do quadro	4	0x02	O usuário pode identificar o pacote com um número arbitrário para conferir a mensagem de *status*. Se for configurado como 0, nenhuma resposta de *status* é enviada.
		Endereço de 64 bits de destino MSB (5-11) LSB (12)	5	0x00	O endereço do rádio remoto para retornar essa resposta.
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x52	
			11	0x2B	
			12	0xAA	
		Endereço de 16 bits de destino (MY) MSB (13) LSB (14)	13	0x7D	Endereço de 16 bits da fonte. Deve ser igual a 0xFFFE se o endereço do destinatário é desconhecido ou a transmissão broadcast.
			14	0x84	
		Status do comando	15	0x00	0 = OK 1 = ERRO 2 = Comando Inválido 3 = Parâmetro Inválido 4 = Falha na Transmissão
		Comando AT	16	0x44	Nome do Comando AT – Dois caracteres ASCII. O comando D2 permite configurar o comportamento do pino D2.
			17	0x32	
		Valor do Parâmetro			Quando presente indica que se deseja alterar o parâmetro do módulo para o valor fornecido.
	Checksum		18	0x39	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*.

Fonte: Adaptado de Digi (2014, p. 119)

2.3.5 Transmissão de dados no modo de operação API

Para realizar uma transmissão de dados, isto é, enviar uma sequência arbitrária de *bytes* entre dispositivos XBee é necessário conhecer, pelo menos, três tipos de pacotes:

a) Pacote de requisição de transmissão (Tipo do quadro 0x10);
b) Pacote de *status* da transmissão (Tipo do quadro 0x88);
c) Pacote com os dados recebidos (Tipo do quadro 0x90).

A Figura 13 ilustra a dinâmica da troca de pacotes durante a transmissão de dados:

Figura 13 - Troca de pacotes para a transmissão de dados

Fonte: Soares (2012, p.38).

Para iniciar a transmissão de uma mensagem deve-se enviar uma sequência de bytes formatada como apresentado na Tabela 6. Neste exemplo, é realizada a transmissão da mensagem “OLA” para um dispositivo XBee de destino com endereço físico 0x0013A2004062588D.

Tabela 6 - Estrutura de um pacote API de requisição de transmissão

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de *bytes* entre o tamanho e o *checksum.* Neste exemplo 17 bytes
		LSB	2	0x11	
	Dado específico do pacote	Tipo do pacote API	3	0x10	Pacote do tipo 0x10 (comando AT local)
		ID do quadro	4	0x01	O usuário pode identificar o pacote com um número arbitrário para conferir a mensagem de *status*. Se for configurado como 0, nenhuma resposta de *status* é enviada.
		Endereço de 64 bits de destino MSB (5-11) LSB (12)	5	0x00	O endereço do rádio remoto para retornar essa resposta.
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x62	
			11	0x58	
			12	0x8D	
		Endereço de 16 bits de destino (MY) MSB (13) LSB (14)	3	0xFF	Endereço de 16 bits da fonte. Deve ser igual a 0xFFFE se o endereço do destinatário é desconhecido ou a transmissão broadcast.
			14	0xFE	
		Raio do *Broadcast*	15	0x00	Número máximo de saltos de uma transmissão broadcast. (0x00 é o máximo possível).
		Opções	16	0x00	Configurado bit a bit, este campo pode habilitar algumas funcionalidades: 0x01 – Desabilita resposta ACK 0x20 – Habilita Encriptação (Se EE=1) 0x40 - Usa o tempo limite de transmissão estendida para este destino.
		Mensagem (dados)	17	0x4F	Mensagem (dados) que serão transmitidos via RF.
			18	0x4C	
			19	0x41	
	Checksum		20	0x	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*.

Fonte: Adaptado de Digi (2013, p.105).

Após o pacote de requisição de transmissão ser enviado, o dispositivo XBee de destino irá fornecer no pino de saída da serial (DOUT) um pacote contendo a mensagem recebida, o endereço XBee. Estes e outros dados são apresentados na

Tabela 7.

Tabela 7 - Estrutura de um pacote API de recebimento de dados

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de *bytes* entre o tamanho e o *checksum.* Neste exemplo 16 bytes
		LSB	2	0x10	
	Dado específico do pacote	Tipo do pacote API	3	0x90	Pacote do tipo 0x10 (comando AT local)
		ID do quadro	4	0x01	O usuário pode identificar o pacote com um número. Se for configurado como 0, nenhuma resposta de *status* é enviada.
		Endereço de 64 bits de destino MSB (5 -11) LSB (12)	5	0x00	Endereço de 64 bits do remetente.
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x62	
			11	0x58	
			12	0x8D	
		Endereço de 16 bits de destino (MY) MSB (13) LSB (14)	3	0xC5	Endereço de 16 bits do remetente.
			14	0xC4	
		Opções	15	0x00	Configurado bit a bit, este campo pode habilitar algumas funcionalidades: 0x01 – Desabilita resposta ACK 0x20 – Habilita Encriptação (Se EE=1) 0x40 - Usa o tempo limite de transmissão estendida para este destino.
		Mensagem (dados)	16	0x4F	Mensagem (dados) que serão transmitidos via RF.
			17	0x4C	
			18	0x41	
	Checksum		19	0xD5	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*.

Fonte: Adaptado de Digi (2014, p.113).

Após a transmissão dos dados, o dispositivo XBee que originou a transmissão recebe um pacote de *status* que informa se a mensagem foi entregue com sucesso e, caso contrário, informa a natureza da falha. A Tabela 8 apresenta a estrutura de um pacote de *status* da transmissão.

Tabela 8 - Estrutura de um pacote API de status de transmissão

	Campos do pacote		Índice	Ex.	Descrição
Pacote API	Delimitador de início		0	0x7E	Obrigatório em todo pacote API
	Tamanho do pacote	MSB	1	0x00	Quantidade de *bytes* entre o tamanho e o *checksum.*
		LSB	2	0x07	
	Dado específico do pacote	Tipo do pacote API	3	0x8D	Pacote do tipo 0x8D (comando AT local)
		ID do quadro	4	0x01	O usuário pode identificar o pacote com um número arbitrário. Se for configurado como 0, nenhuma resposta de *status* é enviada.
		Endereço de 16 bits de destino (MY) MSB (5) LSB (6)	5	0xC5	Endereço de 16 bits do XBee de destino (se a mensagem foi entregue corretamente). Caso contrário esse endereço é igual ao enviado no pacote de requisição de transmissão.
			6	0xC4	
		Quantidade de tentativas de retransmissão	7	0x00	Número de tentativas de retransmissão.
		Status de entrega (Delivery status)	8	0x00	0x00 = Success 0x01 = MAC ACK Failure 0x02 = CCA Failure 0x15 = Invalid destination endpoint 0x21 = Network ACK Failure 0x22 = Not Joined to Network 0x23 = Self-addressed 0x24 = Address Not Found 0x25 = Route Not Found 0x26 = Broadcast source failed to hear a neighbor relay the message 0x2B = Invalid binding table index 0x2C = Resource error lack of free buffers, timers, etc. 0x2D = Attempted broadcast with APS transmission 0x2E = Attempted unicast with APS transmission, but EE=0 0x32 = Resource error lack of free buffers, timers, etc. 0x74 = Data payload too large 0x75 = Indirect message unrequested
		Status de descoberta (Discovery status)	9	0x01	0x00 = No Discovery Overhead 0x01 = Address Discovery 0x02 = Route Discovery 0x03 = Address and Route 0x40 = Extended Timeout Discovery
	Checksum		10	0xE7	0xFF – Soma dos bytes a partir do índice 3 até o byte anterior ao *checksum*.

Fonte: Adaptado de Digi (2014, p.112).·.

2.3.6 Entradas/saídas digitais e analógicas

Os módulos XBee possuem pinos cujo comportamento pode ser configurado através de *software*. Este tipo de interface é denominado GPIO (*General Purpose Input/Output)*. No módulo XBee PRO Series 2, existe 11 pinos configuráveis através de comandos AT, conforme descrito na Tabela 9. Quatro destes pinos podem ser configurados como entradas analógicas (pinos 17 a 20) e todos podem ser configurados como entrada/saída digital.

Tabela 9 - Pinos Configuráveis (GPIO) no Módulo XBee PRO S2

Nome do pino	Número do pino	Comando AT de configuração
CD/DIO12	4	P2
PWM0/RSSIM/DIO10	6	P0
PWM/DIO11	7	P1
DIO4	11	D4
CTS/DIO7	12	D7
ASSOC/DIO5	15	D5
RTS/DIO6	16	D6
AD3/DIO3	17	D3
AD2/DIO2	18	D2
AD1/DIO1	19	D1
AD0/DI0	20	D0

Fonte: Adaptado de Digi (2014, p.95)

Para configurar as funções analógica ou digitais em um ou mais pinos do módulo XBee, o comando de configuração apropriada deve ser enviado com o parâmetro correto. Depois de enviar o comando de configuração, as alterações devem ser aplicadas no módulo para as definições de IO fazerem efeito. A Tabela 10 apresenta alguns parâmetros para comandos AT.

Tabela 10 - Parâmetros para comandos AT de configuração dos pinos GPIO

Valor do Parâmetro	Descrição
0	Desabilitado
1	Reservado para funções especifica de alguns pinos (led associativo, RSSI PWM, etc.)
2	Conversor analógico digital, *Single Ended* de 10 bits. Disponível somente para os pinos 17 a 20
3	Entrada digital
4	Saída digital em nível baixo
5	Saída digital em nível alto
6-9	Funções alternativas disponíveis em alguns pinos (CTS, RTS, etc.).

Fonte: Adaptado de Digi (201, p.95)

- Amostragem de entradas analógicas e digitais

Os módulos XBee ZB tem a capacidade de monitorar e adquirir amostras de suas entras analógicas e digitais. As amostras de I/O podem ser lidas localmente através do modo AT e/ou transmitidos para um dispositivo remoto através do modo API para os dispositivos que executam este firmware.

Existem três formas de se obter amostras de I/O:

a) **Amostragem periódica -** permite um módulo XBee adquirir uma amostra de I/O e transmiti-lo a um dispositivo remoto a uma taxa periódica. Esta taxa é definida pelo comando AT “IR”, que admite valores entre 0x32 e 0xFFFF (50 a 65535 milissegundos), se IR é definido como 0, a amostragem periódica está desativada. O parâmetro DH (*Destination Address High*) e DL (*Destination Address low*) pode ser definido como 0 para transmitir ao coordenador, ou para o endereço de 64 bits do dispositivo remoto (SH e SL);

b) Amostragem por detecção de mudanças - os módulos podem ser configurados para transmitir imediatamente uma amostra de dados sempre que um pino I/O digital monitorado mudar de estado. O comando AT “IC” é uma máscara de bits que pode ser usado para definir quais pinos de I/O digital devem ser monitorados para uma mudança de estado. As amostras de detecção de alteração são transmitidas para o endereço de 64 bits especificado pelo DH e DL**;**

c) **Amostragem por requisição -** o módulo XBee adquire e transmite uma amostra somente quando recebe uma solicitação, realizada a partir do comando AT “IS”. Este comando pode ser enviado para um dispositivo local, ou para um dispositivo remoto usando o quadro de comando remoto API. Quando o comando “IS” é enviado, o XBee realiza uma leitura de todos os seus canais digitais e analógicos habilitados e retorna uma amostra para o dispositivo solicitante. A tabela 11 apresenta os campos de resposta do comando.

Tabela 11 - Organização de uma amostra de canais analógico e digital

Quantidade de *bytes*	Nome	Descrição
1	Conjunto de amostras	Quantidade de conjuntos de amostras no pacote. Sempre igual a 1.
2	Máscara de canais digitais	Indica quais linhas digitais estão habilitadas. Cada bit corresponde a um pino: bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 3 = AD3/DIO3 bit 4 = DIO4 bit 5 = ASSOC/DIO5 bit 6 = RTS/DIO6 bit 7 = CTS/GPIO7 bit 8 = Não utilizado bit 9 = Não utilizado bit 10 = RSSI/DIO10 bit 11 = PWM/DIO11 bit 12 = CD/DIO12 Exemplo: uma máscara com valor 0x0021 indica que as linhas digitais DIO 0 e 5 (Pinos 20 e 15, respectivamente) estão habilitadas
1	Máscara de canais analógicos	Indica quais pinos estão configurados como entradas analógicas. Cada bit representa um canal analógico: bit 0 = AD0/DIO0 bit 1 = AD1/DIO1 bit 2 = AD2/DIO2 bit 3 = AD3/DIO3 bit7 = Tensão de Alimentação (*Supply Voltage*)
Varável	Amostras dos canais digitais e analógicos habilitados	Os primeiros 2*bytes* indicam o estado de todas os canais digitais habilitados. Se nenhum canal digital está habilitado, esses *bytes* são omitidos. Em seguida, cada entrada analógica habilitada retornará 2*bytes* representando o valor de 10 bits do seu conversor AD. As amostras são ordenadas começando em AN0 até AN3, seguido da tensão de alimentação (*Supply Voltage*), caso habilitada.

Fonte: Adaptado de Digi (2014, p. 96).

A resposta ao comando “IS” pode ser recebida de duas maneiras:

a) Se o módulo XBee estiver operando no Modo Transparente (AT), retorna uma lista delimitada pelo símbolo de retorno, contendo os campos da Tabela 10;
b) Quando operando em Modo API, ao receber uma requisição de comando “IS” o módulo XBee responde ao solicitante com um pacote do tipo 0x88 ou 0x97 (Tabela 2 ou Tabela 4) incluindo no campo “parâmetro do comando” a

amostra de todos os seus canais digitais e analógicos habilitados, conforme detalhado na Tabela 10. Um exemplo de amostra é apresentado na Tabela 12.

Tabela 12 - Exemplo de uma amostra de canais analógicos e digitais

Exemplo	Descrição
0x01=0000 0001_b	1 conjunto de amostras
0x0C0C=0000 1100 0000 1100_b	Canais digitais DIO 2,3,10 e 11 habilitados
0x03 = 0000 0011_b	Canais analógicos A/D 0 e 1 habilitados
0x0408 = 0000 0100 0000 1000_b	Estado das entradas digitais: DIO 3 e 10 em nível alto. DIO 2 e 11 em nível baixo
0x03D0= 976_d	Entrada analógica AD0 = 0x03D0 = 976d
0x0124 = 292_d	Entrada analógica AD1 = 0x124 = 292 d

Fonte: Adaptado de Digi (2014, p.97).

2.4 KITS XBEE SENSORS

A família de kits XBee Sensor é um grupo de sensores que implementam o protocolo ZigBee, movidos à bateria e incorporando um módulo XBee e componentes eletrônicos adicionais responsáveis pelo sensoriamento de fenômenos físicos convertidos em sinais analógicos. Em particular, os XBee Sensors permitem a amostragem em tempo real de circuitos eletrônicos com sensores de temperatura, umidade e luz. Estes dados podem ser recuperados e transmitidos através de comunicações sem fio em uma infraestrutura de rede XBee.

2.4.1 Posição dos componentes do XBee ZB L/T/H (XS-Z16-CB2R)

Os nós sensores XBee ZB L/T/H (XS-Z16-CB2R) têm um botão e um LED de indicação de operação, bem como sensores de luz, temperatura e umidade integrados conforme a Figura 14.

Figura 14 - Posição dos componentes do XBee Sensor

Fonte: Adaptado de Digi (2014, p.62).

A Tabela 13 apresenta as funções dos botões e do LED presente no XBee ZB L/T/H (XS-Z16-CB2R)

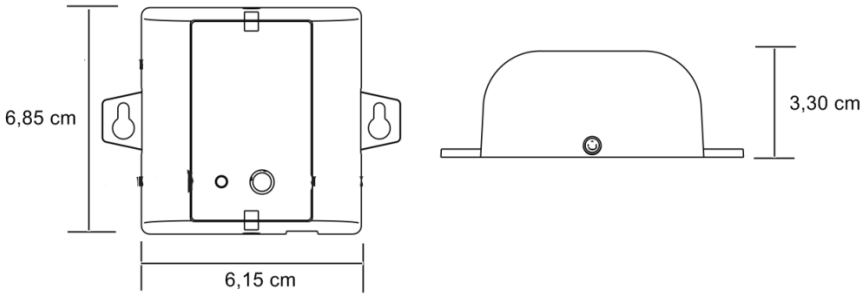
Tabela 13 - Funções dos botões/LED presentes no XBee Sensor, ZB Bat /L/T/H.

Botão/LED	Função
LED Associação/ Ligado	Indica se está ligado e *status* de associação de rede do Sensor XBee. Este LED possui funções diferente dependendo do protocolo RF XBee para o produto.
Botão de reiniciar	Executa a função de reiniciar e múltiplas funções para identificar e configurar o Sensor XBee em uma rede sem fios. Pressione o botão consecutivo dentro de 800 milissegundos para executar cada ação desejada.

Fonte: Adaptado de Digi (2013, p. 62).

Além dos fatores expostos, cabe ressaltar que se caracteriza como um dispositivo de dimensões reduzidas (Figura 15)

Figura 15 - Dimensão do XBee ZB L/T/H (XS-Z16-CB2R)



Fonte: Adaptado de Digi

2.4.2 Características dos sensores XBee ZB L/T/H (XS-Z16-CB2R)

Como o *datasheet* (Digi), os sensores XBee ZB L/T/H (XS-Z16-CB2R) apresentam as características descritas na Tabela 14.

Tabela 14 - Características do datasheet do XBee ZB L/T/H (XS-Z16-CB2R)

Plataforma	XBee® Sensor
Desempenho	
Banda de frequência	ISM 2.4 GHz
Taxa de dados de RF	250000 bps
Faixa interior	40 m
Faixa exterior	120 m
Energia de transmissão	1,25 mW (+1 dBm) modo normal; 2mW (+3 dBm) modo de reforço.
Sensibilidade do receptor (1%)	-96 dBm modo de reforço; -95 dBm modo normal
Sensores integrados	
Sensor de temperatura	Faixa: -18 ° C a +55 ° C (-0,4 ° F a 131 ° F); Precisão: + / - 2 ° C
Sensor de luz do ambiente	Faixa de largura de banda espectral: 360-970 nm (semelhante ao olho humano); Comprimento de onda de pico de sensibilidade: 570 nm
Sensor de Umidade	Faixa: 0 a 100% RH (umidade relativa); Permutabilidade: + / -5% RH (0% RH a 59% HR), + / -8% HR (60% RH a 100% HR); Precisão: + / - 3,5% RH
Vida da bateria	
Ciclo de trabalho / vida útil da bateria estimada	1 leitura por 30 seg / 1,5 anos 1 leitura por minuto / 2,5 anos 1 leitura por hora (ou menos frequente) / 6 anos Suposições baseadas em: ciclos de sono ativado com transmissão único por ciclo ativo; A temperatura ambiente de 21 ° C

Fonte: Adaptado de Digi (2013, p.2).

2.5 TOPOLOGIA DE REDES ZIGBEE

Dependendo da topologia (Figura 6) o envio de um pacote em uma rede ZigBee pode ocorrer por meio de diferentes caminhos existente entre os nós. Assim uma malha ZigBee dispõe de vários caminhos possíveis entre os nós da Rede para a passagem da informação. Desta forma é possível contornar as falhas de um nó que este se torna inoperante, simplesmente mudando o percurso da informação.

A rede ZigBee é dividida em três tipos:

a) ***Mesh* (Malha ou Ponto-a-Ponto):** Na topologia Mesh a rede pode se ajustar automaticamente, tanto na sua inicialização como na entrada ou saídas de dispositivos na Rede. A Rede se auto organiza para otimizar o trafego de dados. Com vários caminhos possíveis para a comunicação entre os nós, este tipo de Rede pode abranger, em extensão, uma grande área geográfica;

b) ***Cluster Tree* (Árvore)**: Semelhante à topologia de Malha, uma Rede em árvore, tem uma hierarquia muito maior e o coordenador assume o papel de nó mestre para a troca de informação entre os nós roteadores e dispositivos finais;

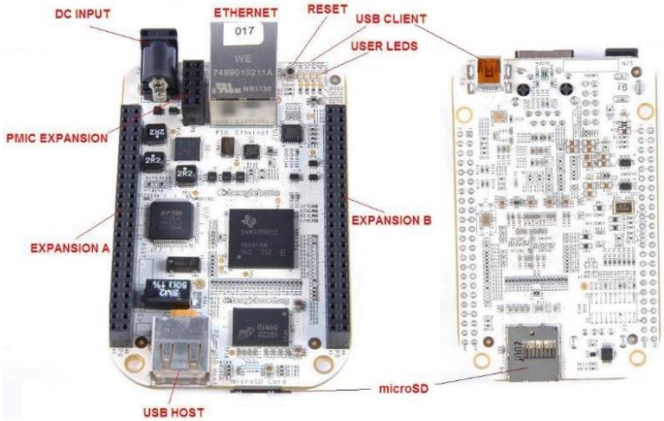
c) ***Star* (Estrela):** É uma das topologias de Rede ZigBee mais simples de serem implantadas, é composta de um nó Coordenador e quantos Dispositivos Finais forem precisos. Este tipo de Rede deve ser instalado em locais com poucos obstáculos à transmissão e recepção dos sinais, como por exemplo, em uma sala sem muitas paredes ou locais abertos, cobrindo uma área limitada

2.6 PLATAFORMAS MICROPROCESSADAS DE BAIXO CUSTO

2.6.1 Beagleboard

Desenvolvido pela *Texas Instrument* juntamente com a *DigiKey*, o *BeagleBoard* é uma plataforma *open hardware* que foi criada com o intuito de difundir o microprocessador de mídia DM3730, que é parte componente da plataforma (BEAGLEBOARD, 2012). O *BeagleBoard* possui 3 modelos que se diferem pelo número de saídas, processadores, memória RAM e tamanho da placa. O modelo básico, nomeado de *BeagleBone*, possui 2x46 pinos (7xADC, 66xGPIO, 2x12C, 5xUART, SPI, CAN, 8xPWM, LCD, *parallel*, MMC/SDIO), conectividade Ethernet e USB, processador AM335x 720MHz ARM Cortex-A8, dentre outras características, conforme mostrado na Figura 16.

Figura 16 - Componentes do *BeagleBone*

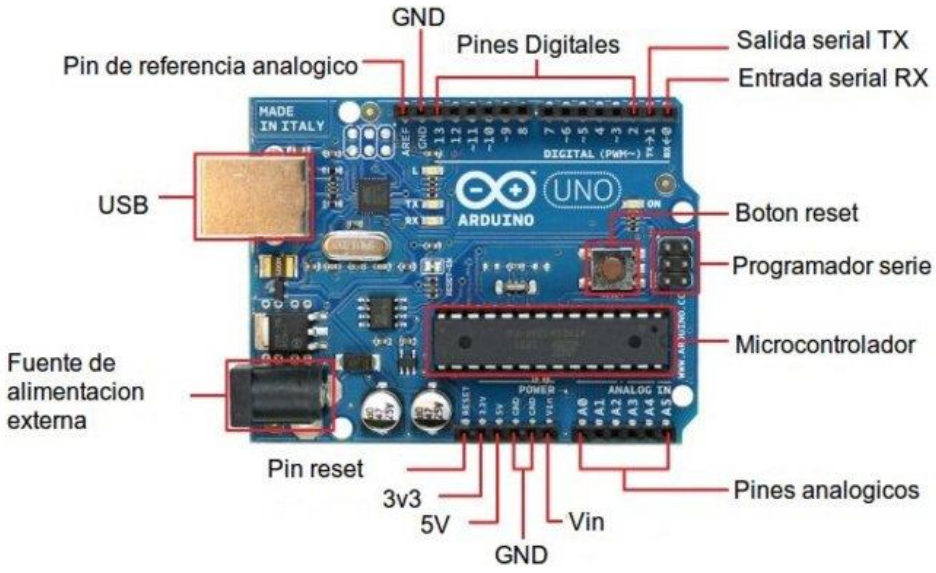


Fonte: BEAGLEBOARD (2012).

2.6.2 *Arduino*

O *Arduino* é uma plataforma *open hardware* de protótipos eletrônicos baseado em software e hardware de fácil uso. Ele é capaz de interagir com o ambiente através de entradas por uma variedade de sensores e assim sendo possível controlar iluminação, motores e outros tipos de atuadores (ARDUINO, 2012). Basicamente, o *Arduino* é composto por um microcontrolador – responsável pelo processamento das informações –, resistores, capacitores, cristal oscilador, entre outros componentes. O microcontrolador está programado por uma linguagem própria do fabricante, que é baseada na linguagem *Wiring* (plataforma semelhante a computação física) e o ambiente de desenvolvimento é baseado na linguagem *Processing*. Há vários modelos de *Arduino* disponíveis no mercado que se diferenciam pela capacidade de processamento do microcontrolador, quantidade de entradas analógicas, entradas e/ou saídas digitais e capacidade das memórias internas. Um deles é o *Arduino* UNO que possui 14 entradas/saídas digitais (das quais 6 podem ser usadas como saídas PWM), 6 entradas analógicas, conexão USB, microcontolador ATmega328, dentre outras características, como mostrado na Figura 17.

Figura 17 - Partes do *Arduino* UNO

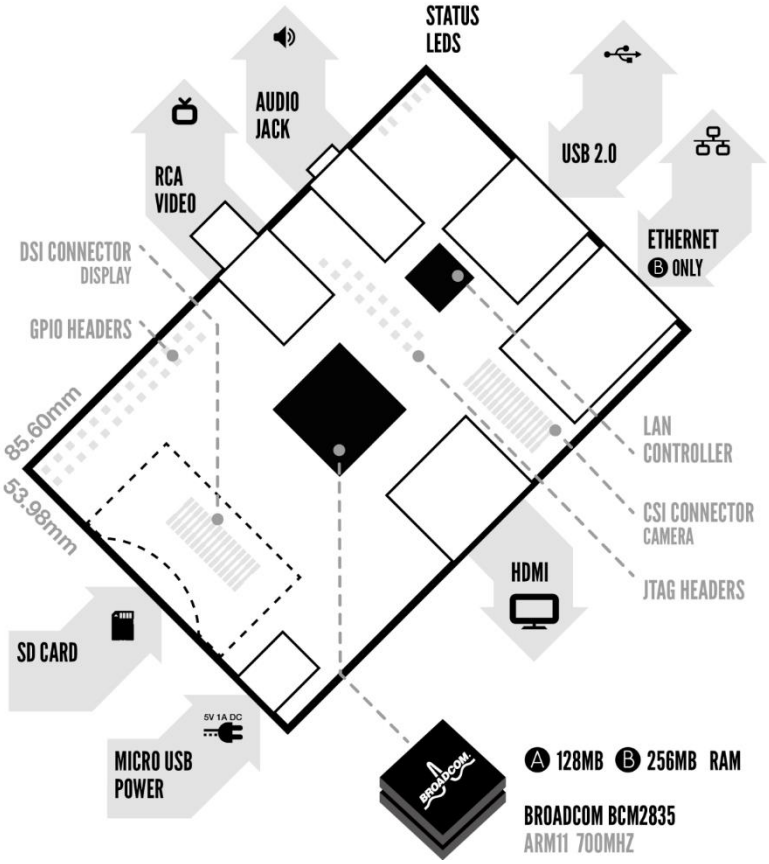


Fonte: COMOHACER.EU (2014).

2.6.3 Raspberry Pi

O *Raspberry Pi* foi criado com o intuito de ser um computador de baixo custo. A plataforma oferece dois modelos (A e B), sendo que o modelo B foi lançado primeiro. Este possui recursos como conectividade Ethernet e USB, entrada para cartão SD, processador 700MHz ARM 1176JZF-S core, 8xGPIO, UART, I²C, SPI, +3.3 V, +5 V, terra, dentre outras características (DENNIS, 2013), conforme podemos ver na Figura 18.

Figura 18 - Componentes do Raspberry PI



Fonte: ELMOONY (2012).

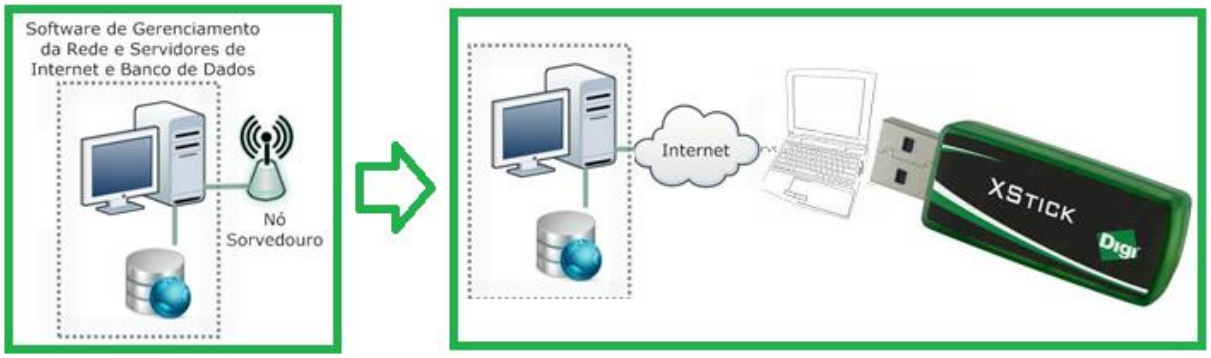
Para o desenvolvimento deste trabalho foi utilizada uma RSSF que utiliza como protocolo de comunicação o padrão ZigBee, composta por coordenador, roteadores e dispositivos finais. São comparados os dois modos de transmissão durante a realização dos testes e também a utilização dos comandos AT no modo de operação API para efetuar a solicitação dos dados coletados pelos roteadores e dispositivos finais. Na amostragem de entradas analógicas e digitais é aplicada a amostragem periódica, onde os roteadores e dispositivos finais enviam seus dados

ao coordenador. Os sensores utilizados são o XBee ZB L/T/H (XS-Z16-CB2R) e a topologia da rede, dependo do objetivo do teste de validação, *Mesh* ou *Star*. Para a criação da RSSF móvel, foi utilizada a plataforma microprocessada de baixo custo Raspberry Pi que atuará como coordenador da rede, onde armazenará internamente os dados captados pelos sensores XBee, para posteriormente enviá-los ao servidor remoto. A escolha desta plataforma se deve ao fato da disponibilização de uma unidade pela COPPE/UFRJ (Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia/Universidade Federal do Rio de Janeiro) devido a parceria da universidade com a Unipampa através do projeto FAPERGS (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul), Desenvolvimento de sistema e método integrado de uso de tecnologias de VANT (Veiculo Aéreo Não Tripulado) e RSSF aplicado ao manejo extensivo em contexto de integração lavoura-pecuária.

3 AMBIENTE DE DESENVOLVIMENTO

Neste capítulo são apresentados os materiais e métodos adotados no desenvolvimento do trabalho. Para tanto, inicialmente são abordadas as ferramentas de *software* e na sequência os elementos de *hardware*.

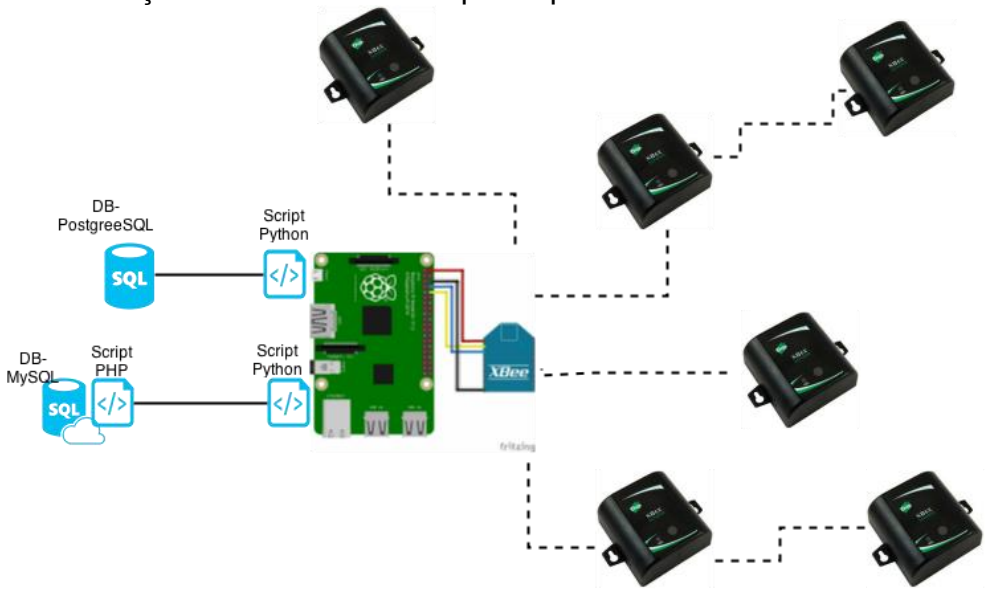
Figura 19 - Modelo para o desenvolvimento da primeira parte do trabalho



Fonte: Própria autora (2014).

O modelo para o desenvolvimento da última parte do trabalho está definido da seguinte forma, os sensores ZB L/T/H enviam os dados coletados para o rádio conectado na *Raspberry Pi* conforme a Figura 20.

Figura 20 - Ilustração dos elementos do protótipo desenvolvido



Fonte: Própria autora (2015).

3.1 *SOFTWARE*

A Digi disponibiliza para o sistema operacional *Windows* a ferramenta de

software X-CTU (Digi, 2015), ilustrada na Figura 7, que possui uma interface intuitiva e de fácil utilização para configuração do *firmware* dos dispositivos, criação da RSSF e captação dos dados coletados pelos sensores. Durante o desenvolvimento da primeira parte do trabalho além desta interface padronizada, foi verificada a necessidade de criar um Software de Gerenciamento da Rede customizado, incluindo a capacidade de agrupar os dados coletados pelo nó Concentrador durante um intervalo de tempo e, periodicamente, enviar para um servidor remoto, por meio da Internet, estes dados. Para tanto, foi utilizada a linguagem de programação *Python* por ser uma linguagem robusta e de fácil utilização, inclusive indicada pela Digi para o desenvolvimento de soluções que envolvam os seus nós sensores (Digi). Para o desenvolvimento da última parte do trabalho foi utilizado o software de gerenciamento da rede como base para a criação dos scripts de captação de dados dos sensores, envio dos dados para o banco de dados local na *Raspberry Pi* bem como o envio para o banco de dados remoto. Também foram desenvolvidos scripts para criação de uma interface para que o usuário possa acessar os dados salvos no banco de dados remoto.

Para o desenvolvimento do *software* da primeira parte do trabalho foram utilizadas as seguintes ferramentas:

a) Python;
b) Qt designer;
c) driver do XStick.

Para a programação em *Python* as bibliotecas utilizadas foram:

a) import serial;
b) from *datetime* import *datetime*;
c) from *binascii* import b2a_hex;
d) from PyQt4 import QtCore, QtGui;
e) import sys;
f) from status import*;
g) from threading import Thread.

Os arquivos “.py” foram salvos da seguinte maneira:

a) stat.py – contém o programa principal que possui a interface gráfica para o usuário;
b) status.py - contém o desenvolvimento da interface gráfica;

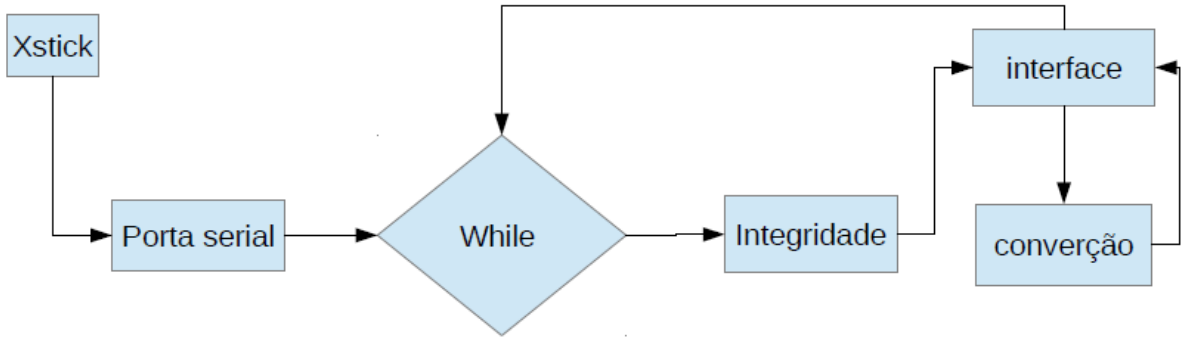
c) servidortcp.py – contém o programa que recebe os dados do programa cliente e os armazena em um arquivo .txt;

d) xbee.py - contém o programa cliente que recebe os dados enviados dos sensores ao XStick e os envia ao servidor.

Primeiramente, para realizar a comunicação serial com o XStick é necessário seguir o tutorial de instalação disponibilizado pela CHIP (2014). Depois de realizada a instalação, foi importada a API serial.py para que fosse utilizada a função Serial para criar a comunicação com a porta em que o XStick está conectado.

O arquivo stat.py possui a seguinte estrutura, conforme a Figura 21.

Figura 21 - Fluxograma do programa stat.py



Fonte: Própria autora (2014)

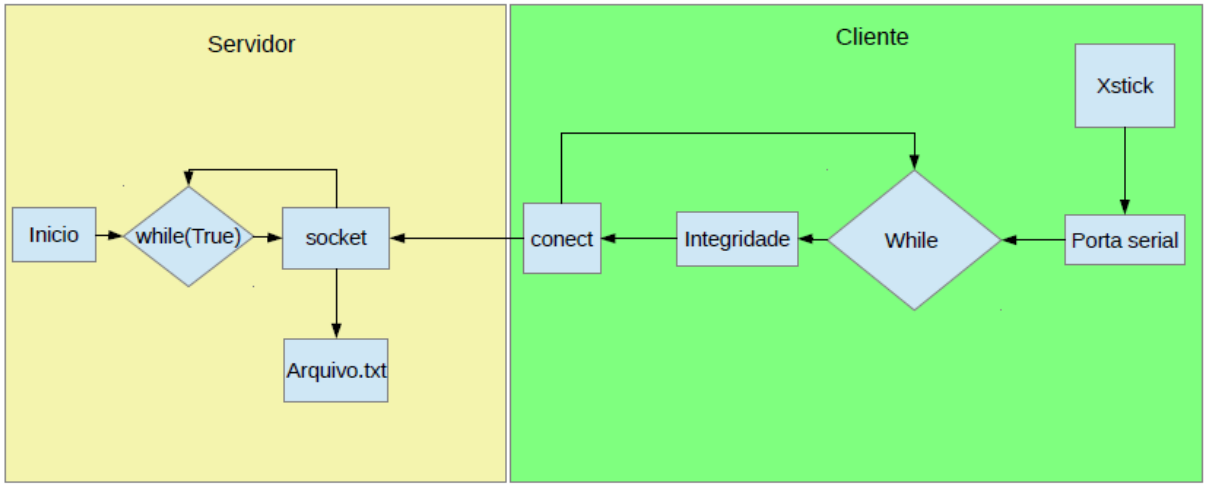
Todo dado que chega à porta serial através do XStick é convertido para hexadecimal com a função “b2a_hex” que pertence a API *binascii*. Em um laço de repetição (*while*), o programa fica escutando a porta aguardando pela chegada de um quadro (frame). Quando um frame chega, é feita uma verificação parcial da integridade do pacote - no momento não está sendo considerada a verificação através do *checksum* - sonde se verifica se o frame que está sendo avaliado inicia com o delimitador “7E”. Após esta verificação é iniciada a identificação do sensor, através da função “updateLCDNumber” com dois parâmetros: o SN (Serial Number) e o frame completo.

Na interface, a partir do SN, os campos de luminosidade, temperatura e umidade pertencentes ao sensor são enviados para o dicionário - lista de valores de referência - à variável “dict” em que o SN está relacionado com a posição do dicionário. No caso de o pacote ser reconhecido é feita uma conversão de hexadecimal para decimal utilizando a função “conv”, que tem como parâmetro os

bytes do frame correspondentes aos campos indicados. Após realizar esta conversão é feito o cálculo para transformar os valores encontrados nas unidades correspondentes. A função plaintText cria o log dos dados captado pelo XStick salvando-o em um arquivo, log.txt, e mostra em tempo real na interface de visualização no campo de texto.

Os arquivos servidortcp.py e o xbee.py possuem a seguinte estrutura, conforme a Figura 22.

Figura 22 - Fluxograma do programa servidotcp.py e xbee.py



Fonte: Própria autora (2014).

O servidor cria um socket para ficar escutando, atualmente sendo capaz de aceitar apenas um cliente por vez. Quando ele recebe uma conexão, o arquivo de log é criado. No momento em que a conexão é fechada por parte do cliente, o arquivo de log é fechado e então o socket fica à espera de uma próxima conexão. O cliente opera em lotes, armazenando pacotes enviados pelos sensores. Após atingir uma quantidade determinada de pacotes, abre uma conexão com o servidor e envia os frames armazenados. Ao concluir este envio, a conexão é fechada e o cliente passa a armazenar mais frames dos sensores para realizar um próximo envio.

Para o desenvolvimento dos *scripts* da última parte do trabalho foram utilizadas as seguintes ferramentas:

a) Python;
b) PHP;
c) JavaScript;
d) JSon;
e) HTML;

f) CSS.

Foi utilizada a ferramenta Python, por oferecer possuir algumas bibliotecas que facilitaram o desenvolvimento do projeto, entre elas:

a) import os, commands;
b) from datetime import datetime;
c) import socket;
d) import psycopg2;
e) import serial;
f) import sys;
g) import json;
h) import urllib2.
i) from xbee import ZigBee;
j) from binascii import b2a_hex.

Cabe ressaltar também o uso de PHP, de JSON e de HTML. A linguagem PHP para realizar os acessos ao banco de dados, devido a maior parte dos servidores HTTP executarem scripts desta linguagem com pouca ou nenhuma configuração, e também por ser a linguagem mais utilizada neste contexto e com isso apresentar uma vasta quantidade de materiais didáticos e tutoriais. O JSON (*JavaScript Object Notation*) para o envio de informações do servidor para o cliente, que é descrito no RFC 4627 como um formato leve para transferência de dados entre o *back-end* e *front-end* e vice-versa. O HTML por ser a linguagem de estruturação de conteúdo padrão de internet, definido pelo W3C (*World Wide Web Consortium*). E o *JavaScript*, por ser a linguagem mais utilizada para a programação WEB, e possuir diversas bibliotecas entre elas a de manipulação dos elementos HTML, CSS, criação de gráficos, entre outros. A biblioteca utilizada foi:

a) jQuery: facilita a manipulação de itens nos documentos html, tratamento de eventos, animação e transferência de dados de forma simples. É amplamente utilizado na Internet, por isso é suportado pela maioria dos navegadores atuais.

Os scripts desenvolvidos foram salvos na *Raspberry Pi* e no servidor da seguinte maneira. Na *Raspberry Pi* estão divididos em coleta e armazenamento local e envio para armazenamento remoto:

a) savedblocal.py - contém o script que coleta os dados enviados pelos nós sensores ao coordenador conectado a *Raspberry Pi* e os armazena no banco de dados (DB – *database*) local;
b) sendtoserver.py - contém o script que busca os dados armazenado no banco de dados local e os envia ao servidor remoto.

No Servidor, os scripts foram divididos em duas categorias: *Back-end* ou *Server-side* onde as atividades são executadas no servidor e *Front-End* ou *Client-side* onde as atividades são executadas no navegador do cliente. No back-end foram criados os seguintes scripts:

a) setamostra.php – contém o script que recebe os dados enviados pelo sendtoserver.py, converte os valores dos sensores e os armazena no banco de dados do servidor;
b) getdata.php – contém o script que conecta com o banco de dados do servidor e seleciona os dados solicitados pelo usuário e retorna em formato JSON;
c) getdatacsv.php – contém o script que conecta com o banco de dados do servidor e seleciona os dados solicitados pelo usuário e retorna em formato JSON;

No front-end foram criados os seguintes scripts:

a) criatabela.js – contém o script que gera a tabela apresentada para o usuário com os dados coletado através do getdata.php;
b) criacsv.js – contém o script que gera o arquivo csv para o usuário salvá-lo em seu computador;
c) relatorios – artigo do J*oomla*™*,* plataforma que permite a criação e gestão de sítios web dinâmicos, que o contém html com o formulário para o usuário escolher um intervalo de visualização dos dados, e os apresenta através do criatabela.js.

Primeiramente, para utilizar a Raspberry Pi como coordenador da rede, é necessário seguir os seguintes passos:

a) formatar o cartão de memória SDCard (4GB ou mais) seguindo o tutorial disponibilizado pelo RANTS & RAVES – THE BLOG! (2013);
b) instalar o sistema operacional, para este trabalho foi utilizado o RASPBIAN mas pode ser utilizado qualquer sistema disponível para essa plataforma,

seguindo o tutorial disponibilizado pelo RASPBERRYPI.ORG;

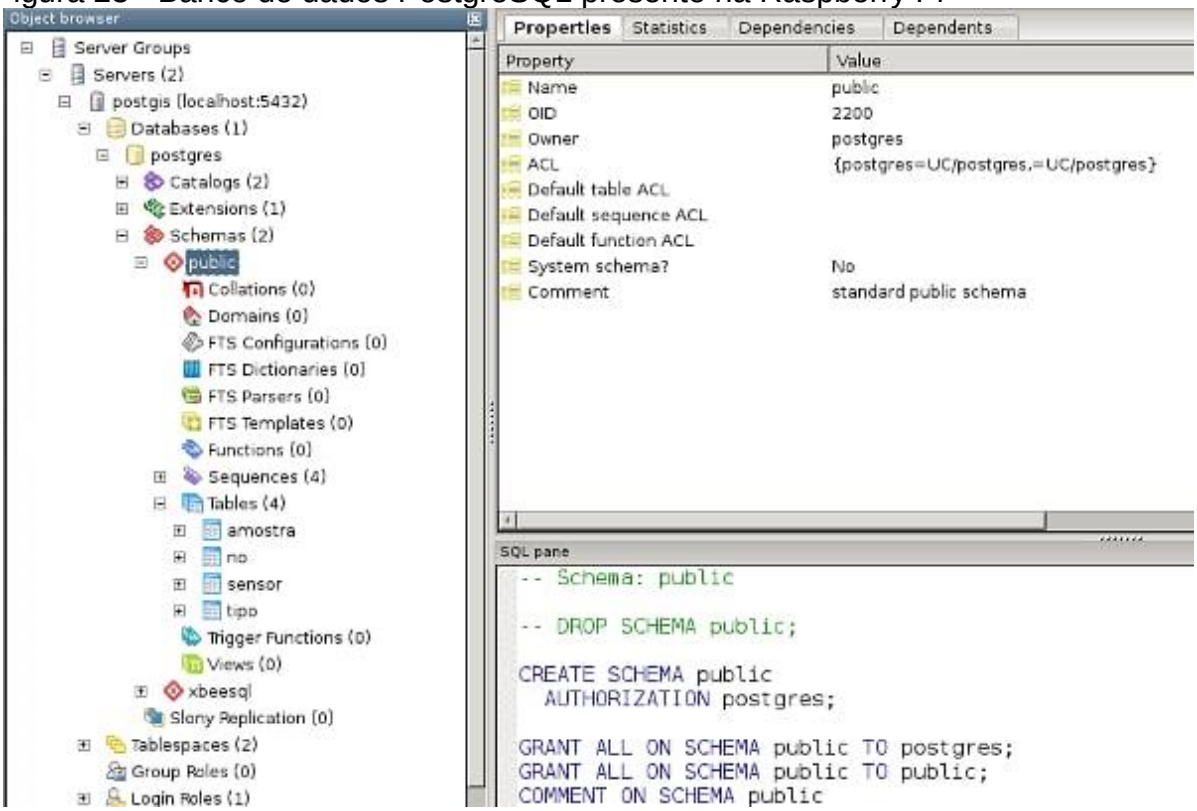
c) é recomendado instalar o gerenciador de pacotes “pip”, para facilitar a instalação dos pacotes Python, para isso foi utilizado o tutorial disponibilizado pelo RASPBERRYPI.ORG;

d) instalar as bibliotecas necessárias para o funcionamento dos scripts desenvolvidos:
   - pip install serialport;
   - pip install xbee;
   - pip install urllib3;
   - pip install simplejson;
   - sudo apt-get install libpq-dev python-dev.

e) instalar o banco de dados PostgreSQL, utilizando o tutorial disponibilizado pelo RASPBERRY PG (2013), juntamente com a ferramenta *PGadmin3* para gerenciamento do banco através de uma interface gráfica.

Após a configuração da Raspberry Pi e a instalação do PostgreSQL, é necessário a criação do DB conforme a Figura 23.

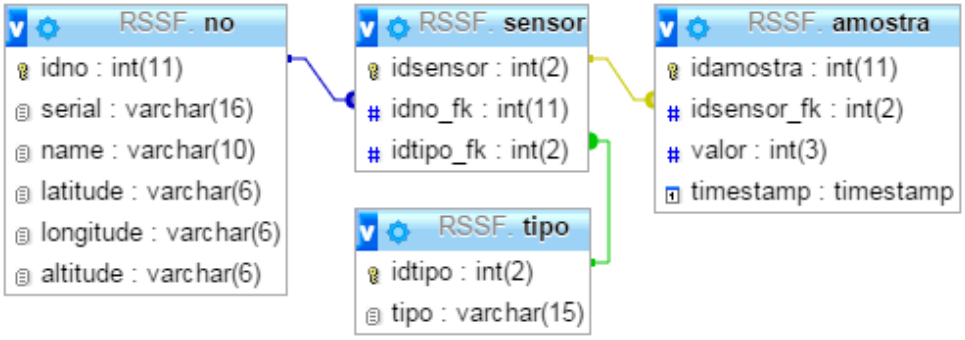
Figura 23 - Banco de dados PostgreSQL presente na Raspberry Pi



Fonte: Própria autora (2015).

O DB local, possui como estrutura as tabelas “amostra”, “no”, “sensor” e “tipo”. Os relacionamentos destas tabelas são apresentados na Figura 24.

Figura 24 - Relacionamento do Banco de dados PostgreSQL



Fonte: Própria autora (2015).

Na tabela “no” o campo “idno” corresponde ao identificador numérico de cada nó no DB, este valor é auto preenchido pelo DB, o campo “serial” corresponde ao *serial number* do nó sensor, o campo “*name*” corresponde ao nome que o usuário define ao nó, conforme a Figura 25, os campos latitude, longitude e altitude corresponde a posição que o nó foi colocado no campo, no momento do desenvolvimento do trabalho estes campos não foram utilizados. Todos estes campos são preenchidos pelo usuário.

Figura 25 - Dados da tabela “no”

	idno [PK] serial	serial character varying(16)	name character	latitude character	longitude character	altitude character
1	1	0013a2004078ecdd	Sensor1			
2	2	0013a2004089bfbf	Sensor2			

Fonte: Própria autora (2015).

Na tabela “tipo” o campo “idtipo” corresponde ao identificador numérico de cada tipo, este valor é auto preenchido pelo DB, o campo “tipo” corresponde a tag do sensor este valor é preenchido pelo usuário, conforme a Figura 26.

Figura 26 - Dados da tabela tipo

	idtipo [PK] serial	tipo character varying
1	1	luminosidade
2	2	temperatura
3	3	umidade

Fonte: Própria autora (2015).

Na tabela “sensor” o campo “idsensor” corresponde ao identificador numérico de cada sensor, este valor é auto preenchido pelo DB, o campo “idno_fk” corresponde a uma chave estrangeira referente ao “idno” da tabela “no” e o campo “idtipo_fk” corresponde a uma chave estrangeira referente ao “idtipo” da tabela “tipo”, os campos “idno” e “idtipo” são preenchidos pelo usuário, conforme a Figura 27.

Figura 27 - Dados da tabela sensor

	idsensor [PK] serial	idno_fk integer	idtipo_fk integer
1	1	1	1
2	2	1	2
3	3	1	3
4	4	2	1
5	6	2	2
6	7	2	3
*			

Fonte: Própria autora (2015).

Conforme a Figura 27 o dado da linha 1, por exemplo, pode ser convertido da seguinte forma: “idno_fk” corresponde ao identificador numérico da tabela “no” com “name” Sensor1 e “idtipo_fk” corresponde ao identificador numérico da tabela “tipo” com “tipo” luminosidade.

Na tabela “amostra” o campo “idamostra” corresponde ao identificador numérico de cada amostra, o campo ”idsensor_fk” corresponde a uma chave estrangeira referente ao “idsensor” da tabela “sensor”, o campo “valor” corresponde ao valor coletado pelo sensor e o campo “timestamp”, com formato em texto data e hora (AAAA-MM-DD hh:mm:ss), corresponde ao momento em que ocorreu a leitura, conforme a Figura 28.

Figura 28 - Dados da tabela Amostra

	idamostra [PK] serial	idsensor_f integer	valor integer	timestamp timestamp without time zone
619	619	6	661	2015-05-30 19:21:42.3816
620	620	7	709	2015-05-30 19:21:42.3816
621	621	4	33	2015-05-30 19:21:47.557995
622	622	6	667	2015-05-30 19:21:47.557995
623	623	7	716	2015-05-30 19:21:47.557995
624	624	4	32	2015-05-30 19:21:52.585167
625	625	6	667	2015-05-30 19:21:52.585167
626	626	7	716	2015-05-30 19:21:52.585167
627	627	4	33	2015-05-30 19:21:57.955674
628	628	6	667	2015-05-30 19:21:57.955674
629	629	7	716	2015-05-30 19:21:57.955674

Fonte: Própria autora (2015).

Após a criação do DB na *Raspberry PI* são executados dois scripts conforme a Figura 20: o primeiro “savedblocal.py” em Python realiza o salvamento dos dados coletado pelo rádio no DB local. Os frames enviados pelos nós são apresentados na Figura 29.

Figura 29 - Frames recebidos dos nós.

```
pi@raspberrypi ~ $ lendo frame
frame:  {'source_addr_long': '\x00\x13\xa2\x00@\x89\xbf\xbf', 'source_addr': ';\x9a', 'id': 'rx_io_data_long_addr', 'sample
s': [{'adc-1': 33, 'adc-2': 660, 'adc-3': 708, 'dio-11': True, 'dio-4': False}], 'options': 'A'}
conectando com o banco de dados
iniciando a conexao
conexao realizada
Done
lendo frame
frame:  {'source_addr_long': '\x00\x13\xa2\x00@\x89\xbf\xbf', 'source_addr': ';\x9a', 'id': 'rx_io_data_long_addr', 'sample
s': [{'adc-1': 34, 'adc-2': 666, 'adc-3': 716, 'dio-11': True, 'dio-4': False}], 'options': 'A'}
conectando com o banco de dados
iniciando a conexao
conexao realizada
Done
lendo frame
frame:  {'source_addr_long': '\x00\x13\xa2\x00@\x89\xbf\xbf', 'source_addr': ';\x9a', 'id': 'rx_io_data_long_addr', 'sample
s': [{'adc-1': 34, 'adc-2': 666, 'adc-3': 716, 'dio-11': True, 'dio-4': False}], 'options': 'A'}
conectando com o banco de dados
iniciando a conexao
conexao realizada
Done
```

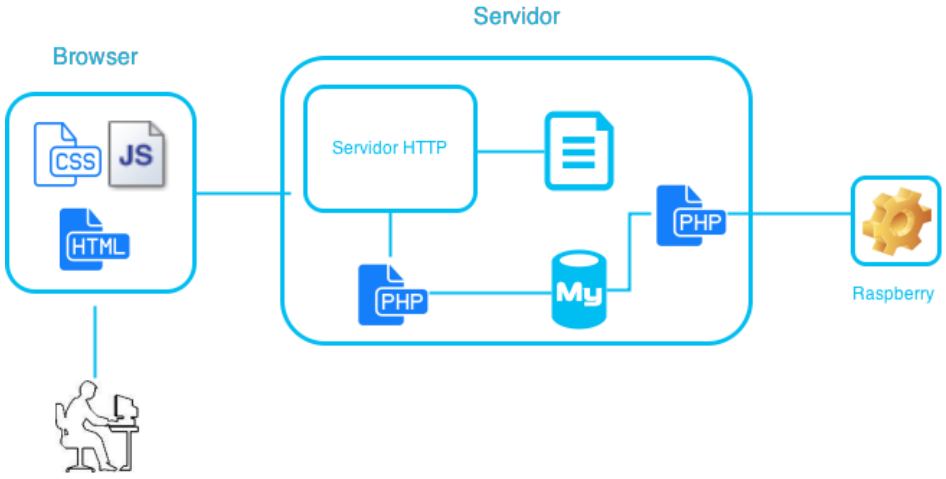
Fonte: Própria autora (2015).

Dentre os campos enviados cabe destacar os relevantes para o presente trabalho: source_addr_long - *Serial Number* do sensor, *sample* – valores captados pelos sensores presentes no nó – sendo adc-1 correspondente aos dados de luminosidade, adc-2 aos dados de temperatura e adc-3 aos dados de umidade.

O segundo script “sendtoserver.py” em Python realiza o envio dos dados do banco de dados local para um banco de dados remoto.

No servidor remoto, conforme a Figura 30, o script “setamostra.php”, em PHP, recebe os dados enviados pela Raspberry Pi, os converte e os armazena no DB MySQL.

Figura 30 - Estrutura do comportamento do servidor remoto



Fonte: Própria autora (2015).

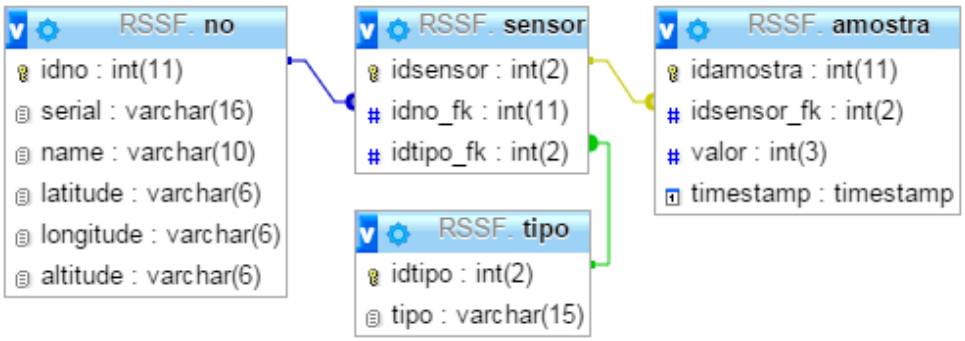
O banco de dados MySQL, é apresentado através da Figura 31.

Figura 31 - Banco de dados MySQL presente no servidor remoto.

Fonte: Própria autora (2015).

Ele possui as mesas tabelas e relacionamento, que o banco de dados PostgreSQL, presente na Raspberry Pi, conforme a figura 32.

Figura 32 - Relacionamento banco de dados MySQL



Fonte: Própria autora (2015).

A estrutura da interface gráfica para visualização e exportação dos dados coletados será apresentada e explicada na seção 3.1.2.

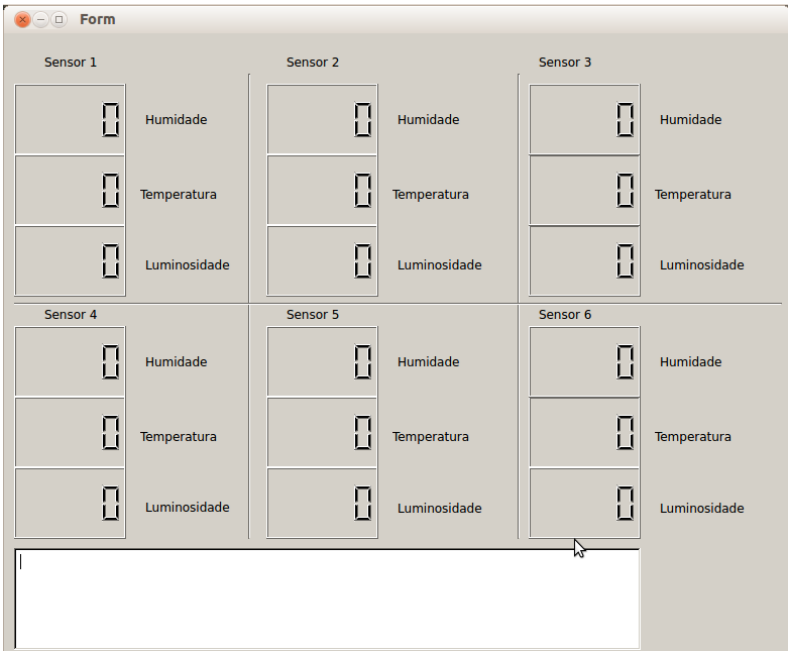
3.1.1 Interface gráfica do Software

Para desenvolver a interface gráfica da primeira parte do trabalho foi utilizada a aplicação *Qt Designer* por ser uma ferramenta de edição de interface gráfica *drag-and-drop*, onde é possível construir a interface gráfica com base no que se está vendo durante o desenvolvimento da interface. Além disso, permite a exportação do resultado final de XML para Python, facilitando a integração com o programa de coleta dos dados dos sensores.

Com a ferramenta *Qt Designe*r, foi montado um grupo contendo 6 conjuntos de dados, cada conjunto de dados é relacionado a um sensor e contendo os 3

campos de dados: luminosidade, temperatura e umidade, conforme a Figura 33, identificados como “lcdNumber_X”, onde o X é o número do campo que está relacionado. Na interface gráfica há um campo que informa o frame que está sendo tradado, naquele instante, este campo está identificado como “logframe”.

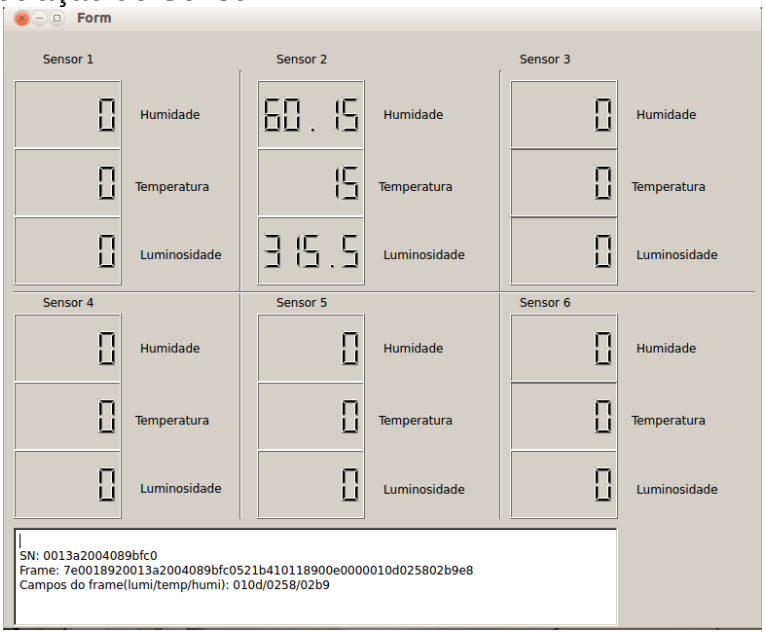
Figura 33 - Tela inicial antes de associar os sensores



Fonte: Própria autora (2014)

Na Figura 34 é possível verificar atualização do “lcdNumber_X” do Sensor2 e atualização do *plaintText* no campo de texto da interface gráfica.

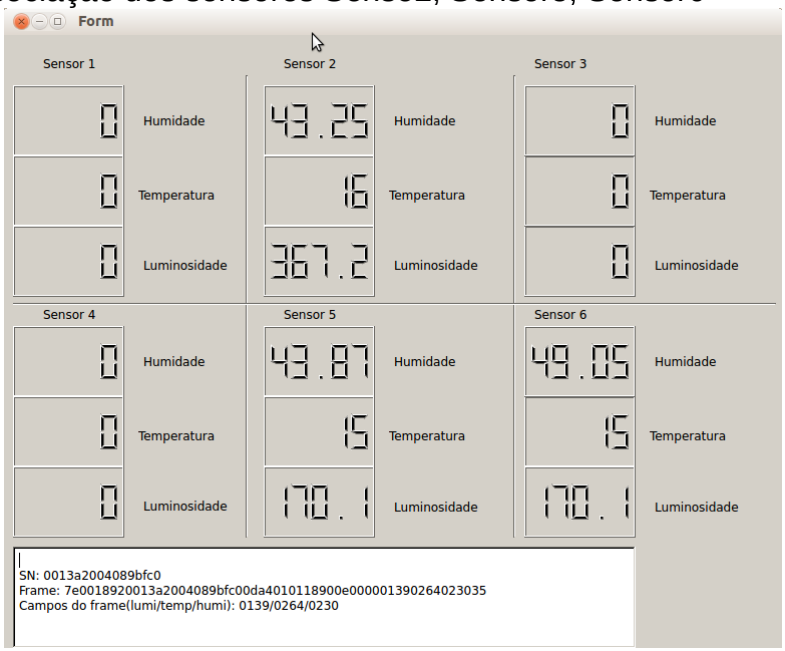
Figura 34 - Associação do Sensor2



Fonte: Própria autora (2014)

A Figura 35 mostra a associação de outros sensores na PAN.

Figura 35 - Associação dos sensores Senso2, Sensor5, Sensor6

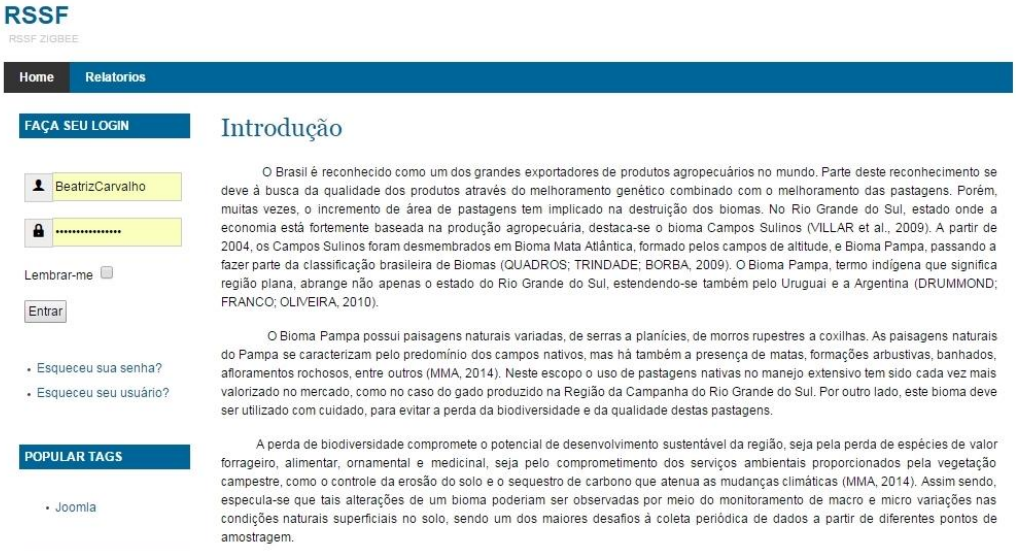


Fonte: Própria autora (2014)

3.1.2 Interface de visualização dos dados do servidor remoto

Para visualizar os dados o usuário acessará a url do servidor remoto, a interface do usuário é criada a partir do Joomla^{TM,} plataforma ou CMS (*Content Management System*) que permite a criação e gestão de sítios web dinâmicos, onde haverá uma aba chamada Home, que contém uma apresentação do trabalho e uma aba chamada Relatorios, conforme FIGURA 36.

Figura 36 - Estrutura da url do servidor remoto



Fonte: Própria autora (2015).

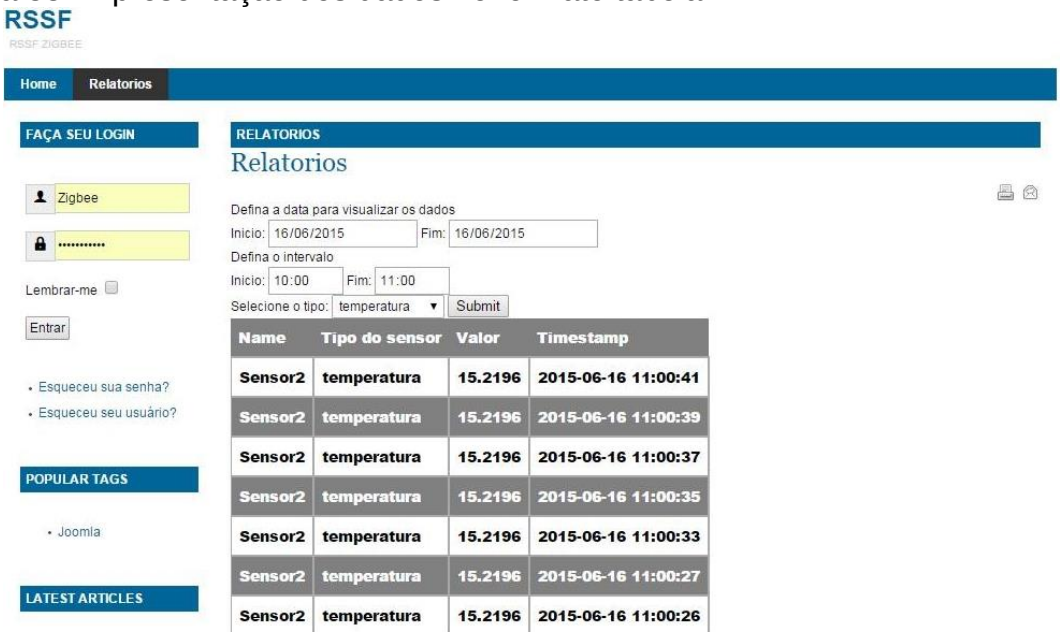
Ao acessar a aba “Relatorios”, o artigo do Joomla™ que contém um formulário html para o usuário escolher um intervalo de tempo, data e o horário, e o conjunto de dados de interesse - luminosidade, temperatura, umidade ou todos eles - para visualização a partir dos dados presentes no banco de dados, conforme a Figura 37.

Figura 37 - Formulário criado através do html do artigo Joomla™.

Fonte: Própria autora (2015).

Ao definir o intervalo de visualização dos dados, é possível escolher duas formas de apresentação dos dados. A primeira ao clicar no botão “*cria tabela*” os valores definidos para o intervalo de visualização são enviados através do método POST do jQuery no script criatabela.js, para o script “getdata.php”. Este conecta com o banco de dados do servidor e seleciona os dados solicitado pelo usuário, e retorna estes codificados em JSON para o script “criatabela.js” que apresenta os dados em formato de tabela no HTML, conforme a Figura 38.

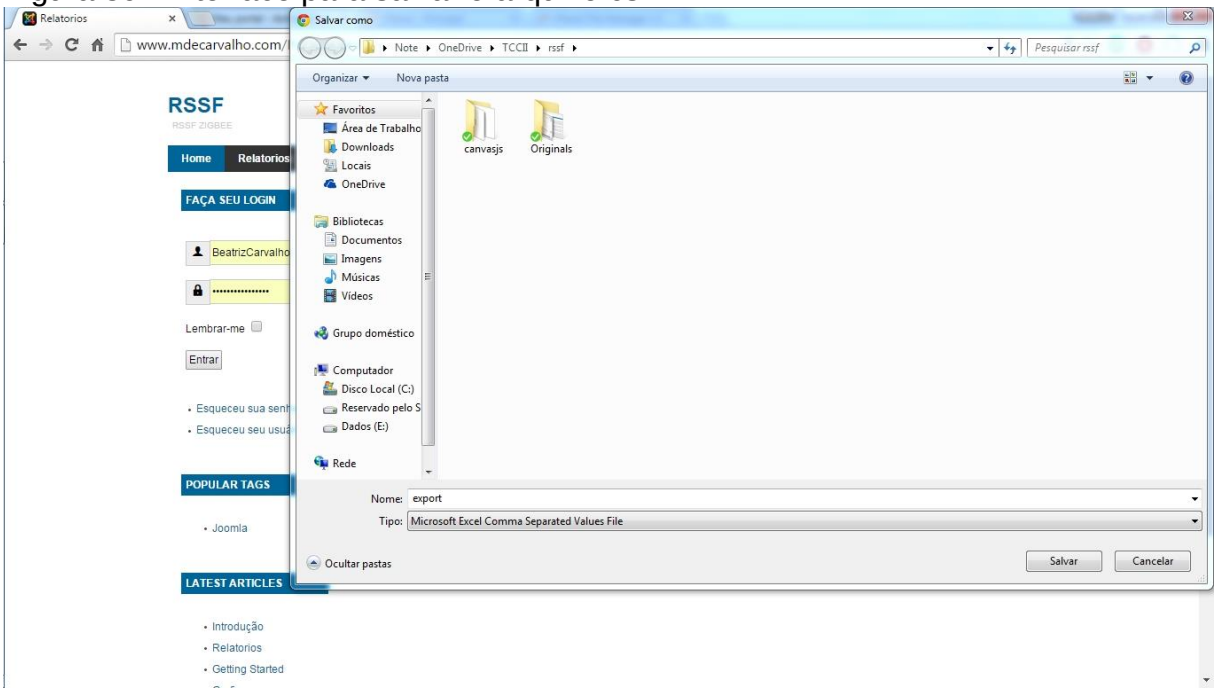
Figura 38 - Apresentação dos dados no formato tabela



Fonte: Própria autora (2015).

Na segunda, ao clicar no botão “cria csv” os valores definidos para o intervalo de visualização são enviados através da função POST, para o script “getdatacsv.php” que conecta com o banco de dados do servidor e seleciona os dados solicitado pelo usuário, e retorna o arquivo contendo os dados em formato csv para o script “criacsv.js”, onde o usuário poderá salvar em seu computador e manipula-lo da maneira que precisar, conforme a Figura 39.

Figura 39 - Interface para salvar o arquivo csv.



Fonte: Própria autora (2015).

A Figura 40 apresenta o formato que os dados selecionados são apresentados no arquivo CSV.

Figura 40 - Formato dos dados do arquivo CSV

```
name,tipo,valor,timestamp
Sensor2,temperatura,13.9293,"2015-06-16 00:30:59"
Sensor2,luminosidade,0,"2015-06-16 00:30:59"
Sensor2,umidade,62.1206,"2015-06-16 00:30:59"
Sensor2,temperatura,13.9293,"2015-06-16 00:30:59"
Sensor2,luminosidade,0,"2015-06-16 00:30:59"
Sensor2,umidade,62.1206,"2015-06-16 00:30:59"
Sensor2,temperatura,13.9293,"2015-06-16 00:30:59"
Sensor2,luminosidade,0,"2015-06-16 00:30:59"
Sensor2,umidade,62.1206,"2015-06-16 00:30:59"
Sensor2,temperatura,13.9293,"2015-06-16 00:30:58"
Sensor2,luminosidade,0,"2015-06-16 00:30:58"
Sensor2,umidade,62.1206,"2015-06-16 00:30:58"
Sensor2,temperatura,13.9293,"2015-06-16 00:30:58"
Sensor2,luminosidade,0,"2015-06-16 00:30:58"
Sensor2,umidade,62.1206,"2015-06-16 00:30:58"
Sensor2,umidade,62.1206,"2015-06-16 00:30:57"
Sensor2,temperatura,13.9293,"2015-06-16 00:30:57"
Sensor2,luminosidade,0,"2015-06-16 00:30:57"
```

Fonte: Própria autora (2015).

3.2 HARDWARE

Para o desenvolvimento deste trabalho, foram utilizados sensores fabricados pela empresa *Digi Internacional.*

3.2.1 Coordenador

Conforme a seção 2.6 o coordenador é o responsável pela RSSF, no desenvolvimento da primeira parte deste trabalho o sensor utilizado como coordenador foi o XStick da Digi, como indica a Figura 41.

Figura 41 - XStick coordenador da rede

Fonte: Digi

Que é um adaptador USB para *XBee Wireless Personal Area Network* (WPAN, Rede pessoal sem fio), fornecendo conectividade local para redes sem fio.

O XStick funciona como uma interface entre a rede XBee e o computador, com uma taxa de transmissão de 1200 bps a 115.2 Kbps. Através deste adaptador é possível a configuração de rede local, diagnóstico ou monitoramento dos dispositivos.

Na segunda parte deste trabalho o sensor utilizado como coordenado foi o módulo XBee PRO 2 conectado com a Raspberry Pi, conforme a Figura 42.

Figura 42 - Modulo XBee Pro2 conectado na Raspberry Pi

Fonte: Própria autora (2015).

3.2.2 Roteadores e Dispositivos Finais

Para atuarem como roteadores e dispositivos finais foram utilizados os sensores XBee ZB L/T/H (XS-Z16-CB2R) conforme apresentado na seção 2.4.1, com os *firmwares* configurado através do programa X-CTU, bem como coletas de dados realizadas em parte por este programa e, posteriormente, por meio da aplicação Python desenvolvida para este trabalho.

A Figura 43 apresenta o sensor XBee ZB L/T/H (XS-Z16-CB2R), fora do invólucro original, onde é possível verificar os componentes do sensor. O “a” indica o módulo XBee PRO 2, o “b” indica o LED de associação, o “c” indica o sensor de luminosidade, o “d” o sensor umidade e o “e” o sensor de temperatura.

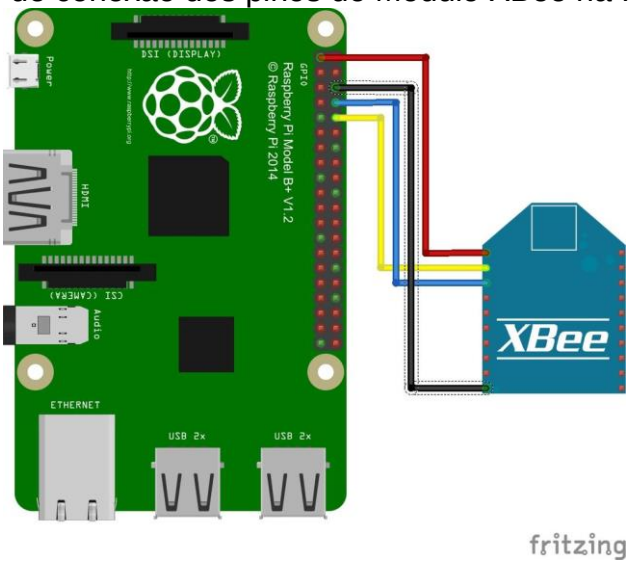
Figura 43 - Sensor XBee ZB L/T/H (XS-Z16-CB2R)

Fonte: Própria autora (2014).

3.2.3 Estrutura da RSSF

Como apresentado na seção 3.2.1 o coordenador da RSSF é um módulo XBee, este foi conectado na Raspberry Pi utilizando o esquemático apresentado na Figura 44.

Figura 44 - Esquemático de conexão dos pinos do Módulo XBee na Raspberry Pi



Fonte: ELMOONY (2012).

Os nós utilizados na RSSF são apresentados na seção 3.2.2.

4 ANÁLISE EXPERIMENTAL

Neste capitulo são apresentados os testes preliminares realizados para identificar a viabilidade do emprego dos sensores XBee ZB L/T/H (XS-Z16-CB2R) para a captação contínua de dados sobre a variação existente na umidade, temperatura e luminosidade incidentes no solo de uma área nativa.

Para tanto, foram definidos diferentes ambientes experimentais com o intuito de conhecer a variação de tensão de operação, a precisão dos dados coletados e a capacidade de cobrir uma grande área. Mais precisamente, os testes são realizados visando um conhecimento empírico mais aprofundado, que ultrapasse as características expostas no *datasheet* do kit de sensores acima referenciado.

Por exemplo, no teste de variação de tensão o objetivo verificar a variação de tensão, onde se aplica a tensão conhecida para o funcionamento dos sensores e esta é reduzida gradualmente para assim conhecer o limite de operação dos sensores, permitindo, em conjunto com os parâmetros de consumo do sensor nos seus três diferentes modos de operação (*sleep*; recepção; amostragem/transmissão) e com as características da bateria, predizer o período de tempo em que o sensor será capaz de operar a campo sem necessidade de substituição de bateria.

Já nos testes de verificação da precisão dos dados coletados, é dado foco na medição da temperatura, incluindo o processo de calibração por meio do ajuste do *fudge* fator a partir do valor real da temperatura do ambiente media paralelamente por equipamentos de alta precisão. Complementarmente, são apresentados experimentos que demonstram a eficácia da topologia em *mesh* para ampliação da área de cobertura da rede, permitindo desta forma cobrir áreas de campo com dezenas de hectares. Por fim, são expostos resultados preliminares e de operação da aplicação desenvolvida para agrupamento e envio de dados agrupados para o servidor remoto responsável pelo armazenamento de longo prazo dos dados, bem como os dados finais após sua implementação física e operacional.

4.1 HARDWARE

4.1.1 Testes nos nós sensores

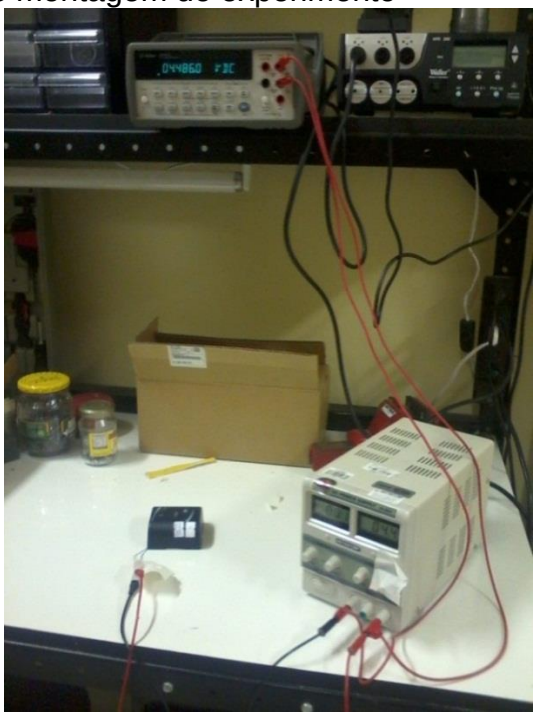
- Teste de variação de tensão nos sensores XBee

Este teste tem como objetivo averiguar o comportamento do sensor conforme a tensão da bateria vai se exaurindo com o decorrer do tempo. Com isso é possível identificar até que momento pode-se confiar na integridade dos dados obtidos em uma determinada aplicação.

Os materiais utilizados foram:

a) um nós sensor XSTICK2 ZB com o *firmware* de Coordenador e interface USB (Figura 45);
b) uma fonte de alimentação DC Instrutherm, modelo FA-3003, com capacidade máxima de 32 volts e 3 amperes;
c) um multímetro digital Agilent, modelo 34401ª, utilizado como voltímetro.
d) um nó sensor XBee ZB L/T/H (XS-Z16-CB2R) com o *firmware* de Dispositivo Final AT e alimentado pela fonte de alimentação; (Figura 45);

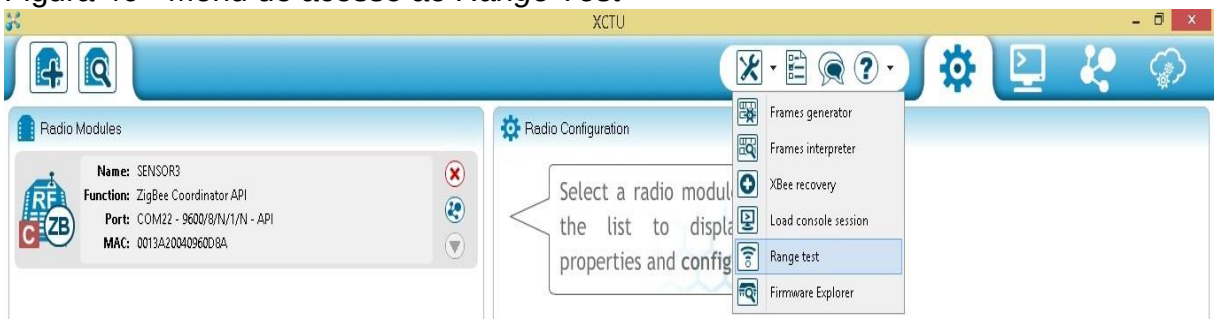
Figura 45 - Esquema de montagem do experimento



Fonte: Própria autora (2014).

e) Um notebook com Sistema Operacional Windows 7 executando o software X-CTU versão 6.1.0, utilizando a opção Range Test presente no menu Tools (Figura 46);

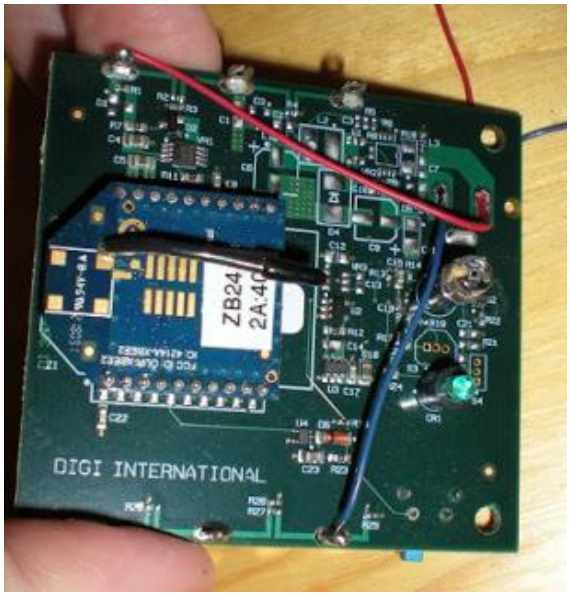
Figura 46 - Menu de acesso ao *Range Test*



Fonte: Própria autora (2014)

Para a realização do teste, foi necessário adaptar os nós sensores XBee ZB L/T/H para alimentação por meio da fonte de tensão. Para tanto, conforme a Figura 47, soldando-se nos polos positivo e negativo dois fios que serviram como conectores da fonte.

Figura 47 - adaptação do sensor XBee



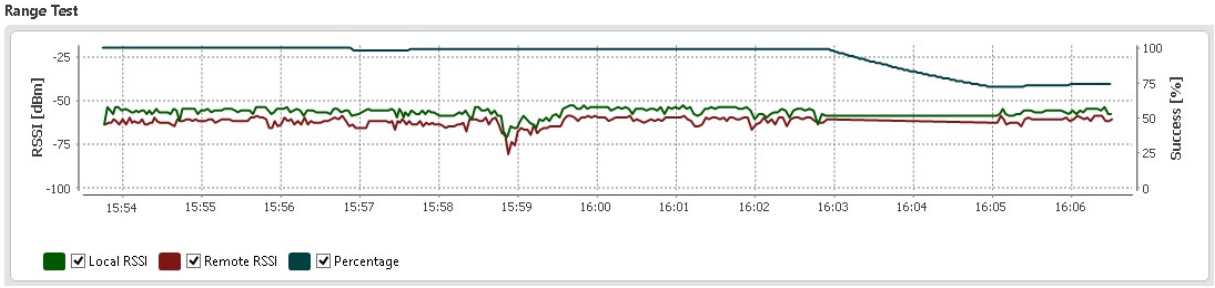
Fonte: Linse (2012).

Em paralelo com a fonte foi colocado o voltímetro de precisão para verificar a tensão real com a qual o sensor estava sendo alimentado, conforme a Figura 26. O sensor (End Device) foi colocado a uma distância de 3,315 m do XStick. O experimento foi iniciado às 15 horas e 13 minutos com a criação da PAN, sucedida pelo início da troca de frames de dados entre os dispositivos. O Dispositivo Final foi

alimentado inicialmente com uma tensão de 4,54 volts e esta foi sendo reduzida de 0,2 em 0,2 volts até chegar à tensão na qual o sensor se desligou: 1,76Volts. Portanto, foi possível identificar que um nó XBee ZB L/T/H, com firmware End Device, é capaz de operar com tensão mínima de 1,8Volts.

Com o auxílio do X-CTU foi possível obter o gráfico da intensidade dos sinais recebidos oriundos do XStick e do sensor, juntamente com a porcentagem de dados perdidos e taxa de erro disponível por meio da ferramenta de obtenção do *ranger* (estimativa de alcance) presente no programa, conforme a Figura 48.

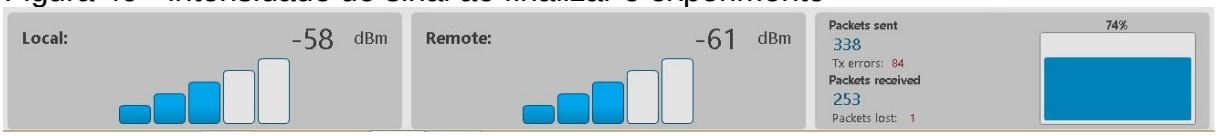
Figura 48 - Gráfico da intensidade do sinal, conforme variação da tensão aplicada no sensor



Fonte: Própria autora (2014).

O gráfico gerado pelo X-CTU indica o sinal dos módulos, tal que o sinal Verde (Local) representa o XStick, o sinal vermelho (Remote RSSI) indica o End Device e o sinal azul (Percentagem) a quantidade de pacotes. A Figura 49 apresenta os valores da intensidade do sinal obtidos ao finalizar o experimento.

Figura 49 - intensidade do sinal ao finalizar o experimento



Fonte: Autor (2014).

A precisão da tensão fornecida pela fonte de alimentação é de +/- 1% + 2 dígitos, conforme o fabricante (Instrutherm), e a precisão da medição da tensão pelo voltímetro é de 0,0035% para CC e 0,06% para CA. Quando a fonte foi configurada para tensão 0 Volts, o voltímetro marcou 6,4 mV.

Os dados obtidos durante o experimento são apresentados na Tabela 15. Pode-se notar que o comportamento do sinal do sensor foi normal na maior parte da variação de tensão, bem como foi possível observar que o sensor opera com uma tensão mínima de aproximadamente 1,8 Volts - menor que isso, se desliga; ao retornar a uma tensão de 1,8 Volts, o sinal retorna, porém, há uma demora porque o

sensor está estabelecendo conexão novamente com a rede (processo de associação – *Join*), conforme explicado no Capítulo 2.

Tabela 15 - Dados obtidos no teste de tensão

Tensão (V)	Comportamento do sinal
4,45	Normal
4,30	Normal
4,06	Normal
3,80	Normal
3,61	Normal
3,42	Perda de Pacote
3,22	Normal
3,01	Normal
2,84	Normal
2,63	Sinal cai consideravelmente
2,46	Sinal estabiliza
2,25	Normal
2,05	Normal
1,87	Sinal oscila bastante
1,76	Perda de pacote começa a ter uma grande taxa de erro e o sensor desliga
1,80	Após alguns instantes o sinal volta e a operação normaliza

Fonte: Própria autora (2014).

- Teste de precisão da temperatura obtido pelos sensores XBee

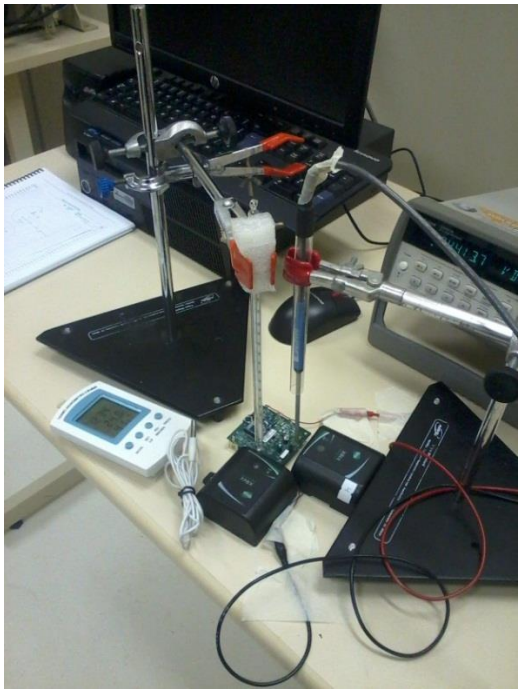
Este teste teve como objetivo averiguar a precisão com que o sensor de temperatura integrado aos sensores XBee coleta os dados.

Os materiais utilizados foram:

a) um nós sensor XSTICK2 ZB com o *firmware* de Coordenador e interface USB (Figura 41);
b) um notebook com Sistema Operacional Windows 7 executando o software X-CTU versão 6.2.0;

c) uma fonte de alimentação DC Instrutherm, modelo FA-3003, com capacidade máxima de 32 volts e 3 amperes;
d) um multímetro digital Agilent, modelo 34401ª, utilizado como voltímetro;
e) um gaussímetro *Cryotronics* 45DSP *Gaussmetter*.
f) três nós sensores XBee ZB L/T/H (XS-Z16-CB2R) com o *firmware* de Router/Dispositivo Final AT, dois alimentado pela fonte e um alimentado por três pilhas alcalinas AA (Figura 50);

Figura 50 - Disposição os sensores acrescentado o sensor alimentado à pilha para a coleta de dados

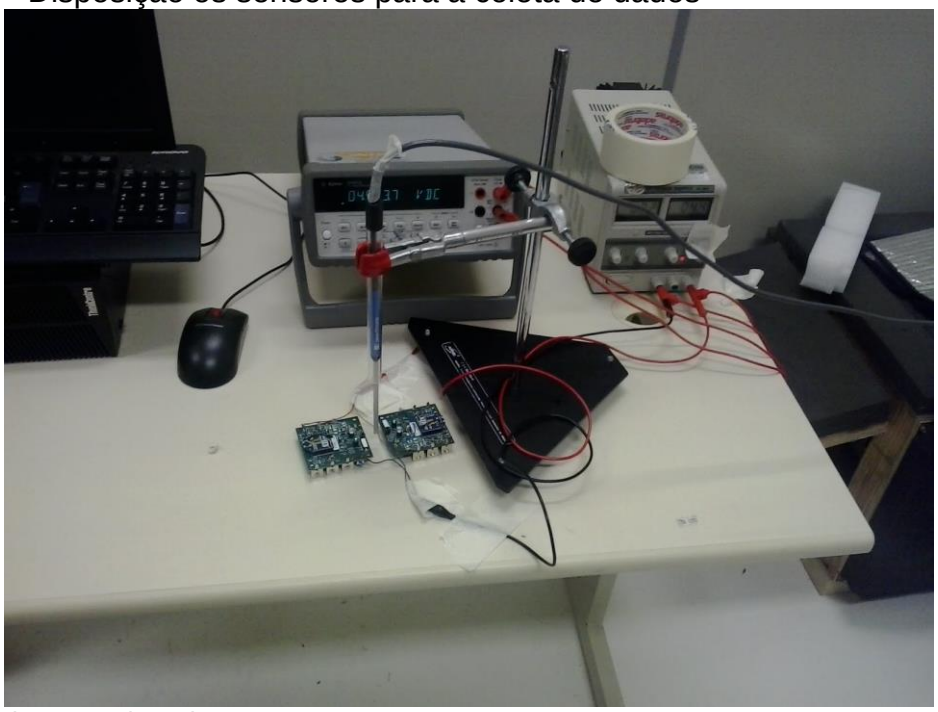


Fonte: Própria autora (2014).

No primeiro teste de verificação de precisão da temperatura foram utilizados dois sensores adaptados para utilizar a fonte de tensão no sensor conforme a figura 47. No segundo teste, foi acrescentado um terceiro sensor, alimentado por pilhas, conforme Figura 50. Em paralelo com a fonte foi colocado o voltímetro de precisão para verificar a tensão real com a qual o sensor estava sendo alimentado.

No primeiro teste os sensores (End Device) foram posicionados conforme a Figura 51.

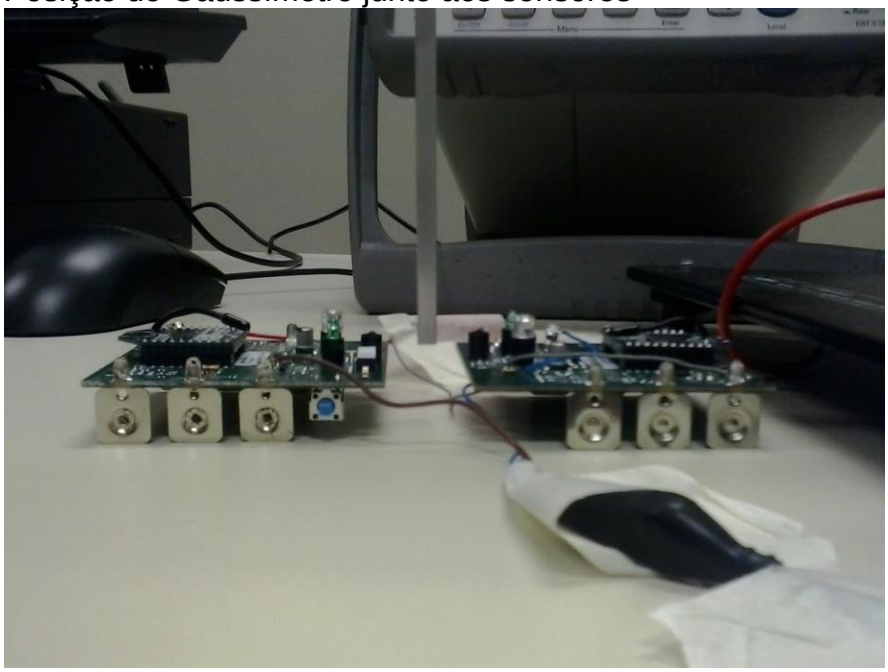
Figura 51 - Disposição os sensores para a coleta de dados



Fonte: Própria autora (2014).

Note que foi feita a retirada do invólucro original, permitindo aproximar a ponta do medidor de campo magnético (Gaussímetro), o qual possui um medidor de temperatura de alta precisão, próxima do sensor de temperatura presente no XBee Sensor (Figura 52).

Figura 52 - Posição do Gaussímetro junto aos sensores



Fonte: Própria autora (2014).

Na Figura 50 é apresenta a posição dos sensores (End Device) no segundo

teste, bem como os outros termômetros que foram acrescentados para a verificação complementar da temperatura.

No primeiro teste, iniciado às 10 h e 22 min, foi iniciada a comunicação entre os dispositivos no X-CTU e acionado ao mesmo tempo o software responsável pela captura dos dados coletados pelo Gaussímetro, representado um processo de coleta paralelo ao realizado pelo X-CTU em relação aos dados enviados pelos sensores. Ao iniciar o experimento a temperatura da sala era de 18,87ºC, segundo o Gaussímetro. Após alguns minutos foi ligado o aparelho de ar-condicionado da marca Elgin, programado para elevar a temperatura até 32ºC. O experimento foi executado por aproximadamente 4 horas e 30 minutos, de modo que às 14h 51 min finalizou-se a captura de dados por meio do software do Gaussímetro e no X-CTU.

No segundo teste, o qual começou exatamente às 10h, foi inicializada a comunicação entre os dispositivos no X-CTU e acionados, ao mesmo tempo, o software para captura dos dados coletados pelo Gaussímetro e o X-CTU. Ao iniciar o experimento, a temperatura da sala era de 18,3ºC, segundo o Gaussímetro. Após alguns minutos foi ligado o aparelho de ar-condicionado mencionado no experimento anterior, também ajustado para aquecer a sala até a temperatura de 32ºC. A duração foi de aproximadamente de 5 horas e 30 minutos, encerrando-se às 15h e 21 min, momento no qual a captura de dados, tanto pelo software do Gaussímetro como pelo X-CTU, foi encerrado.

Neste experimento é empregado o Gaussímetro, o qual possui uma precisão eletrônica de ±0,7ºC, conforme o fabricante *Cryotronics*. Por esse motivo este equipamento foi considerado a referência para a aferição do sensor XBee. Lembrando que o sensor de temperatura presente no XBee Sensor possui uma precisão de ±2ºC, segundo o *datasheet* da DIGI, apresentado na Tabela 14. Assim como no primeiro experimento, os resultados obtidos neste segundo experimento possuem a mesma estimativa para precisão de tensão na fonte de alimentação e de medição de tensão pelo voltímetro.

Para o primeiro teste, os dados coletados no Gaussímetro precisaram ser tratados de modo ao seu processo de amostragem. Mais precisamente, foi realizada uma conversão da referência de tempo para poder indicar a variação da temperatura em um período de tempo medido em segundos, considerando-se que o equipamento relaciona a temperatura medida com o nº do pacote enviado, enquanto que o equipamento lê em torno de 150 pacotes por segundo. Em virtude disto, para

fazer a conversão é utilizada a Equação 2, tal que a col (A) é onde se encontra os pacotes coletados, o 16845 s é o período de tempo de coletado dos dados (fornecido pelo programa Agilent VEE Pro), e o 51873 é a quantidade de pacotes ao final da execução do teste. O gráfico obtido com os dados coletados com o gaussímetro pode ser verificado na Figura 53.

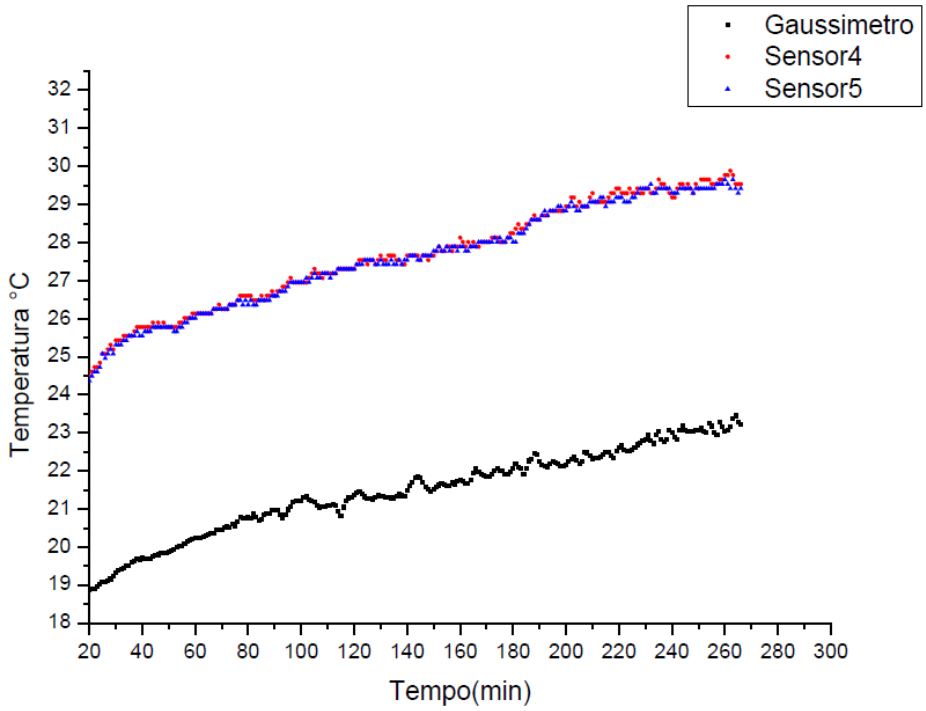
$$\boldsymbol{tempo(s)} = \frac{(\boldsymbol{col(A)} * \mathbf{16845})}{\mathbf{51873}} \qquad \text{Equação 2}$$

Para os dados coletados pelos sensores através do X-CTU, também é necessária uma conversão dos dados coletados em uma representação para a temperatura em graus Celsius. Os dados são coletados e quantizados em 8 bits e representados por meio de dois dígitos hexadecimais. Em função disto, primeiramente é necessário converter o campo responsável pelo dado da temperatura para a representação decimal. De posse deste valor, é aplicada a Equação 3, fornecida pelo fabricante do sensor (Digi), onde ADC3 é o valor lido no sensor convertido em decimal.

$$\left.\begin{aligned} tempc &= \frac{mVanalog - 500}{10} \\ mVanalog &= \left(\frac{ADC3}{1023.0}\right) * 1200 \end{aligned}\right\} \qquad \text{Equação 3}$$

Por meio do gráfico obtido com os dados coletados dos sensores XBee através do X-CTU, o qual pode ser verificado na Figura 53, pode-se perceber uma discrepância nos valores obtidos em relação aos dados coletados no gaussímetro. Dados estes que foram homogeneizados, tal que para cada dado do XBee foi feita uma média do valor obtido para cada conjunto de 180 dados obtidos com o gaussímetro.

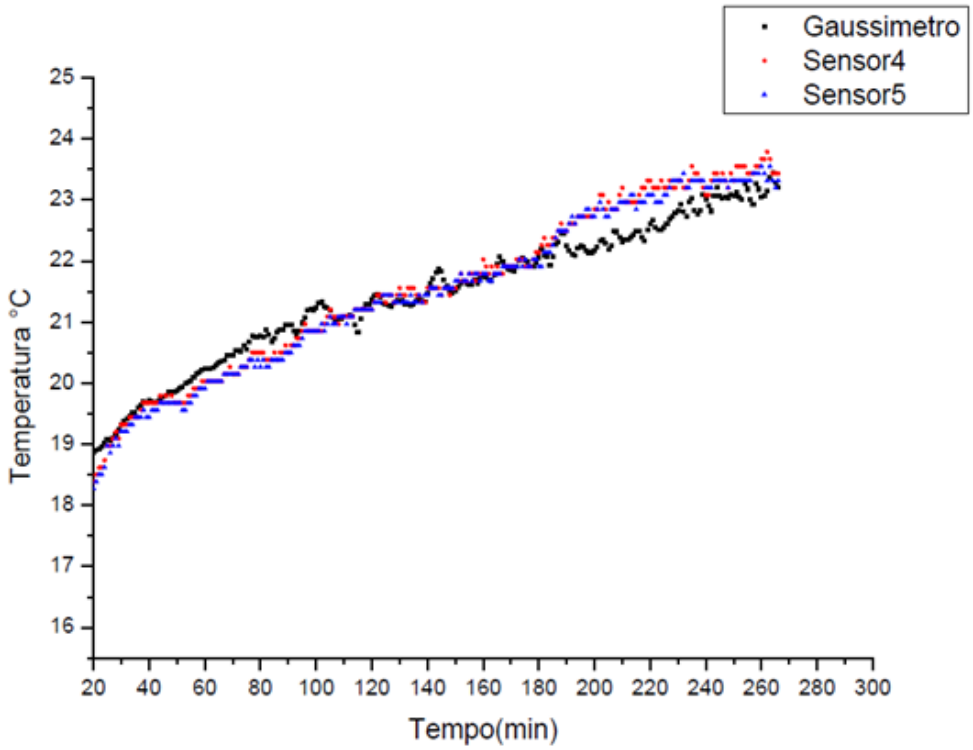
Figura 53 - Gráfico com dados coletados pelos Gaussímetro e pelos sensores XBee utilizando a constante 500



Fonte: Própria autora (2014).

Embora discrepantes, as curvas apresentaram significativa similaridade na tendência de crescimento. Devido a isso, foi realizado um processo de ajuste na equação para obter resultados mais próximos dos dados obtidos pelo Gaussímetro. O processo foi feito variando-se a constante 500 até encontrar o valor que, ao mesmo tempo, minimizasse a diferença de temperatura e a variância desta diferença. A Figura 54 apresenta esta correção, obtida a partir da definição do valor ideal para a constante: 561.

Figura 54 -Gráfico com dados coletados pelos Gaussímetro e pelos sensores XBee utilizando a constante 561



Fonte: Própria autora (2014)

Com o ajuste da equação foi possível obter os dados da Tabela 16, que indica a variação da diferença dos dados do XBee em relação ao gaussímetro. Pode-se perceber que a correção obteve êxito para estes valores, pois a média da diferença foi de 0,26946 para o Sensor4 e 0,26473 para o Sensor5, ou seja, os valores obtidos pelos sensores foram significativamente próximos do valor real.

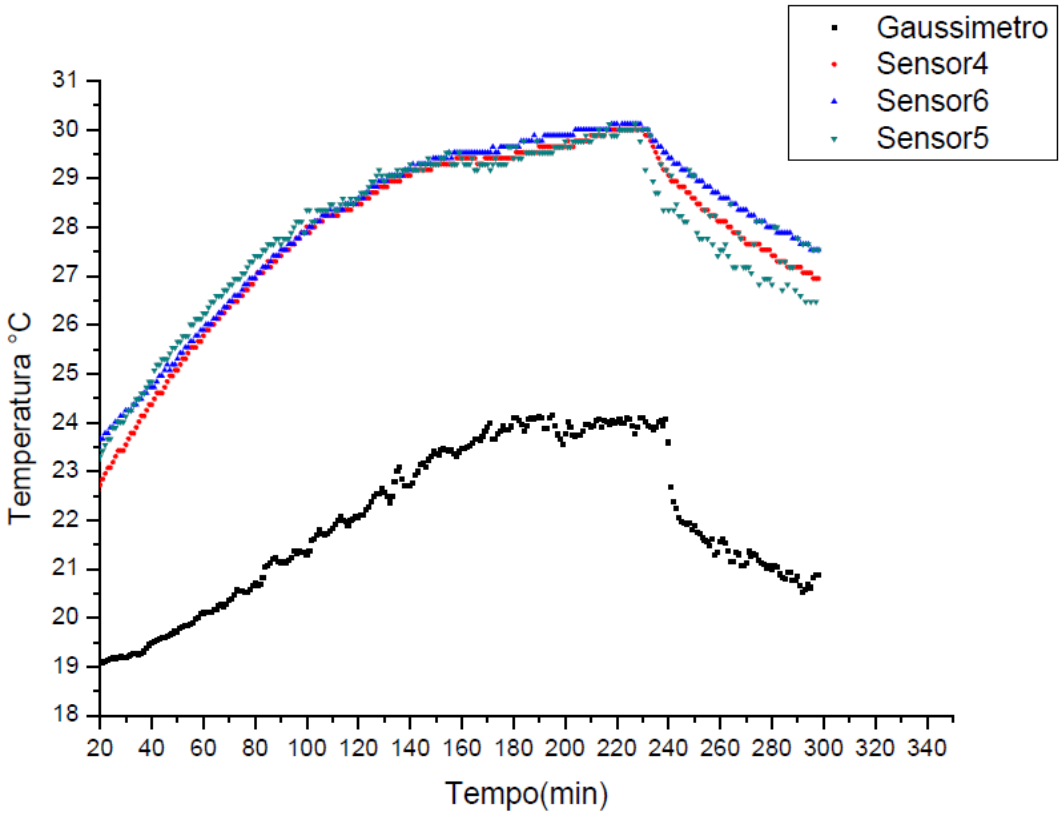
Tabela 16 - Estatística obtida da diferença dos dados obtidos pelo gaussímetro com o sensor XBee

	Gaussímetro – Sensor 4	Gaussímetro – Sensor 5
Mean	0,26946	0,26473
Standard Deviation	0,21104	0,1854
Lower 95% CI of Mean	0,24301	0,2415
Upper 95% CI	0,29591	0,28797
Variance	0,04454	0,03437
Minimum	5,13267E-4	5,13267E-4
Median	0,2401	0,23538
Maximum	0,88272	0,70597

Fonte: Própria autora (2014)

Para confirmar a validade do ajuste na equação de temperatura, foi planejado um segundo teste, buscando verificar se ela se comportaria da mesma maneira. Assim sendo, no segundo teste, também foi necessário realizar a conversão da referência de tempo dos dados obtidos com o Gaussímetro, por meio da Equação 2. Neste experimento, o tempo foi de 20762,4 s, conforme fornecido pelo programa Agilent VEE Pro, e a quantidade de dados coletados foi de 58897 ao final da execução do teste. O gráfico obtido com os dados coletados do Gaussímetro pode ser verificado na Figura 55. De forma análoga, para os dados coletados pelos sensores através do X-CTU foi realizada a conversão necessária aplicando-se a Equação 3. O gráfico obtido com os dados coletados pelo X-CTU pode ser verificado na Figura 55.

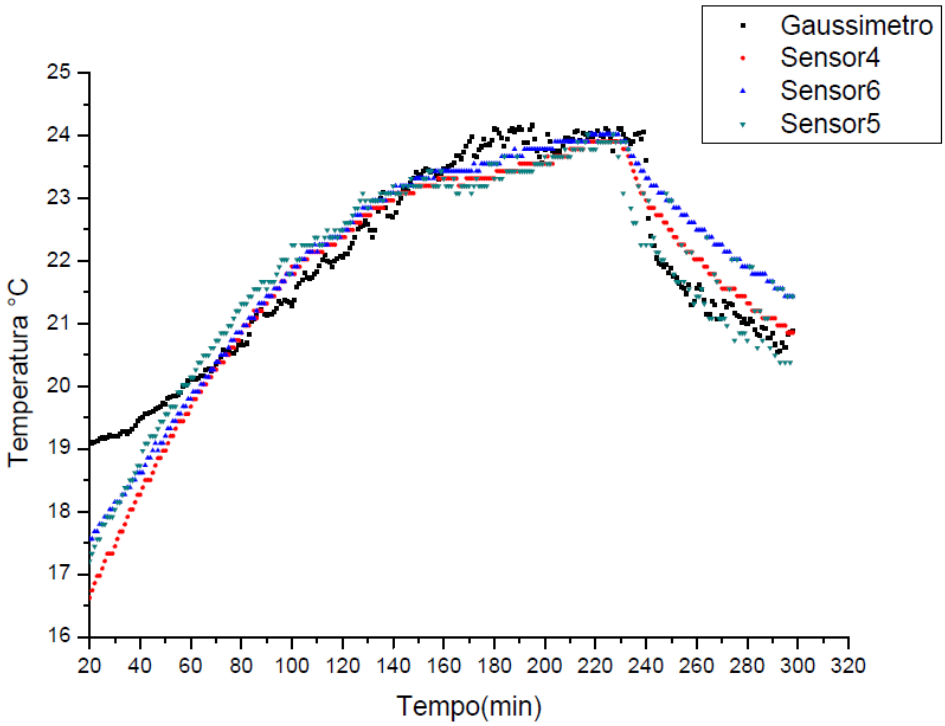
Figura 55 - Gráfico com dados coletados pelos Gaussímetro e pelos sensores XBee utilizando a constante 500



Fonte: Própria autora (2014)

Após ser obtido o gráfico da Figura 34, foi aplicada a equação com a correção obtida no primeiro experimento, para assim validá-la como a mais adequada para a conversão dos valores obtidos pelo sensor X-CTU. O gráfico resultante é apresentado na Figura 56.

Figura 56 - Gráfico com dados coletados pelos Gaussímetro e pelos sensores XBee utilizando a constante 561



Fonte: Própria autora (2014)

Com a aplicação da equação corrigida foi possível obter os dados da Tabela 17, os quais indicam a variação da diferença dos dados do XBee em relação ao gaussímetro. Nota-se que a correção obteve um êxito similar ao do caso anterior, pois a média da diferença foi de 0,45449 para o Sensor4 e 0,044544 para o Sensor5 e de 0,46817 para o Sensor6.

Tabela 17 - Estatística obtida da diferença dos dados obtidos pelo gaussímetro com o sensor XBee

	Gaussímetro – Sensor 4	Gaussímetro – Sensor 6	Gaussímetro – Sensor 5
Mean	0,45449	0,46817	0,44544
Standard Deviation	0,44608	0,39109	0,36158
Lower 95% CI	0,40192	0,42208	0,40282
Upper 95% CI	0,50707	0,51426	0,48805
Variance	0,19899	0,15295	0,13074
Minimum	0,00152	0,00224	0,00386
Median	0,31555	0,33757	0,37589
Maximum	2,48235	1,54393	,89584

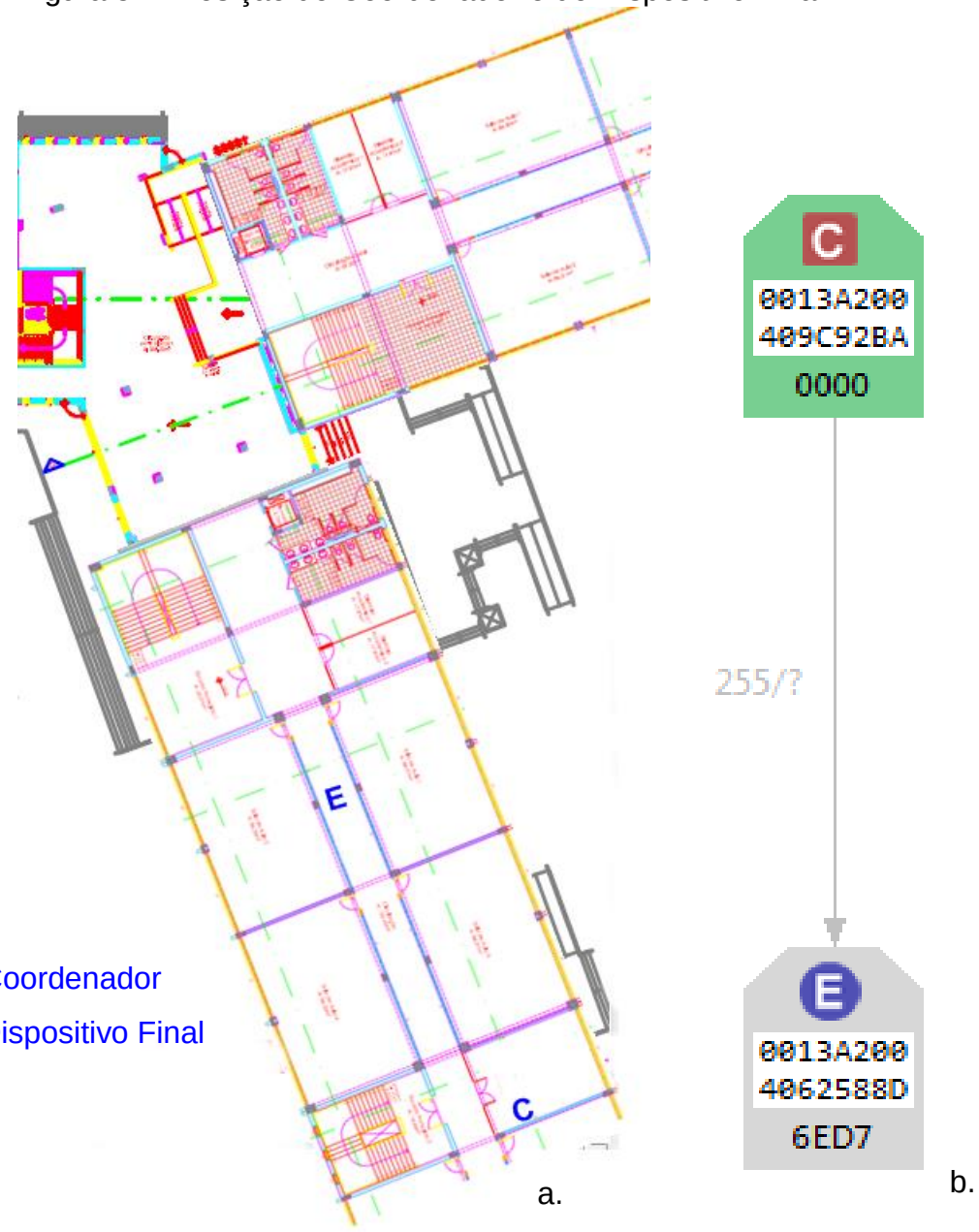
Fonte: Própria autora (2014)

Portanto, os valores obtidos pelos sensores estão significativamente próximos do valor real, mesmo que o processo de homogeneização dos dados entre os dois processos de medição seja caracterizado como uma causa complementar de imprecisão.

4.1.2 Área de cobertura da rede de sensores sem fio

Este experimento teve por objetivo verificar o funcionamento da RSSF no modo *mesh,* pois os sensores utilizados vem de fábrica com *firmware* de *End Device* L/T/H. Na primeira parte do experimento foi utilizado como coordenador da rede o sensor XStick e o sensor XBee ZB L/T/H (XS-Z16-CB2R) como *end device.* Os nós sensores foram configurados para trabalharem com uma potência média, para assim notarmos mais rápidamente a perda de sinal. Nos experimentos, o coordenador ficou fixo na sala 2307, da Unipampa Campus Bagé, localizada no bloco 2, terceiro andar. O *End Device* foi carregado até um ponto onde ocorresse perda de sinal, conforme ilustrado na Figura 57.a. Ao posicionar o *End Device* no local indicado, foi possível obter através do software X-CTU o mapa da RSSF, apresentado pela Figura 57.b.

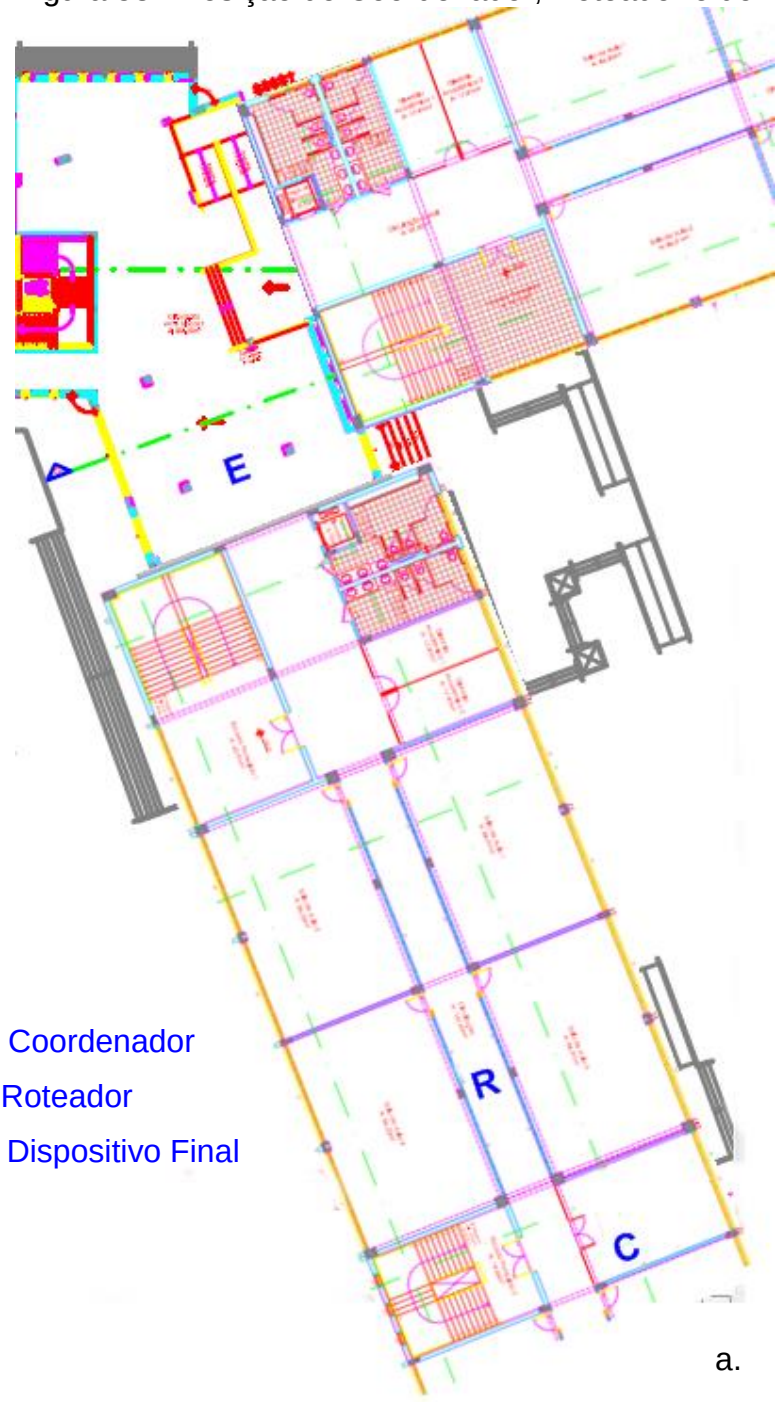
Figura 57 - Posição do Coordenador e do Dispositivo Final



Fonte: Própria autora (2014).

Na segunda parte do experimento foi acrescentado um XBee ZB L/T/H como *router* próximo ao local onde o En*d Device* perdeu o sinal e o end device foi novamente carregado para um ponto com distância suficiente para gerar perda de sinal, conforme apresentado na Figura 58.a. O resultado deste cenário foi o mapa da RSSF ilustrado na Figura 58.b.

Figura 58 - Posição do Coordenador, Roteador e do Dispositivo Final

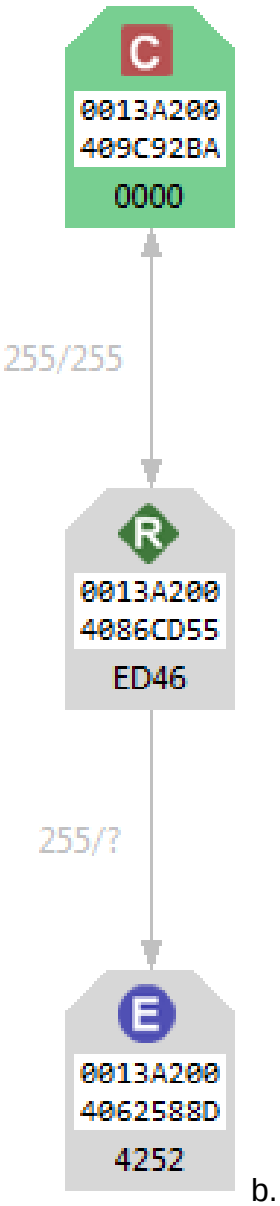


C – Coordenador

R - Roteador

E – Dispositivo Final

a.

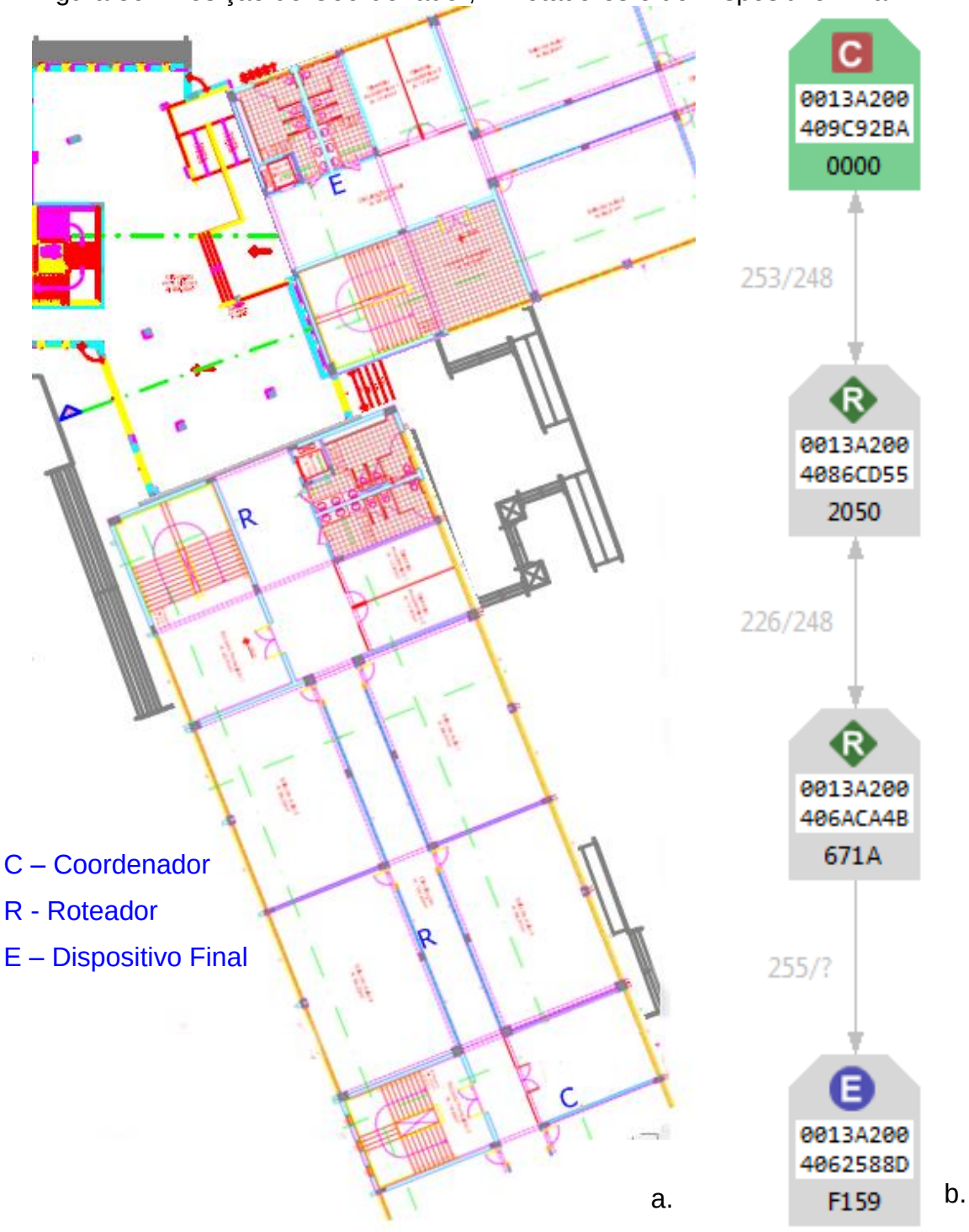


b.

Fonte: Própria autora (2014)

Na última parte do experimento foi acrescentado um segundo sensor XBee ZB L/T/H como *router* próximo ao local onde o *end device* havia perdido o sinal e o end device foi novamente carregado até que ocorresse nova perda de sinal (Figura 59.a), resultando no da RSSF expresso na Figura 59.b

Figura 59 - Posição do Coordenador, 2 Rotadores e do Dispositivo Final



Fonte: Própria autora (2014)

4.1.3 Transmissão dos dados

Para a transmissão de dados do coordenador da rede, foi criada uma aplicação cliente-servidor com base em sockets TCP.

O cliente executa no computador ao qual está conectado o concentrador.

(XStick). Basicamente, a operação do cliente envolve um laço para recebimento e armazenamento temporário em uma estrutura de dados em memória de uma quantidade de pacotes enviados pelos sensores, com *timestamps*. Após a obtenção de uma quantidade parametrizada de pacotes, o cliente abre uma conexão com o servidor e envia os frames armazenados e seus *timestamps* da sua memória para o servidor. Concluído o envio a conexão é fechada, a estrutura de dados é esvaziada e o *cliente* retorna ao seu laço de agrupamento de dados.

O servidor tem o seguinte comportamento: cria um *socket* que aguarda por conexões na porta 5000. Este servidor aceita apenas um cliente por vez. Quando recebe uma conexão, gera um arquivo de log e executa um laço para recepção de frames, armazenando cada frame lido do socket no arquivo. Este processo persiste até o momento em que a conexão é finalizada por parte do cliente, quando o arquivo de log é fechado e o *socket* retorna à espera de uma próxima conexão.

4.1.4 Testes do Protótipo com Sensores em Ambiente Externo

Como parte final do trabalho, foram conduzidos testes práticos com o protótipo desenvolvidos (Seção 3.2.3) em um ambiente externo. O coordenador da rede foi disposto em um ambiente interno, onde este foi alimentado pelo cabo usb conectado ao computador e conectado a uma rede local Ethernet através do cabo UTP, como apresentado na Figura 41. O roteador e dispositivo final utilizados na RSSF foram alimentados por pilhas AAA e dispostos em um ambiente externo, na cidade de Campinas/SP em lugares com vegetação diversificada simulando a RSSF em campo nativo a fim de testar o funcionamento da mesma, conforme as Figuras 60 e 61.

Figura 60 - Disposição do sensor em ambiente externo.

Fonte: Própria autora (2015).

Figura 61 - Disposição do sensor em ambiente externo

Fonte: Própria autora (2015).

Os sensores ficaram expostos às intempéries durante alguns dias, sendo periodicamente executado o script savedblocal.py para coletar e armazenar os dados gerando o log, conforme a Figura 62.

Figura 62 - Log do armazenamento dos dados no DB local

```
2015-06-15 20:27:02.385431 frame:
SN: 0013a2004089bfbf
Campos do frame(lumi/temp/humi): 0/600/712
2015-06-15 20:27:02.707829 insertlocaldata: conexao com banco local realizada
2015-06-15 20:27:03.640692 insertlocaldata: Save data from local database
2015-06-15 20:27:03.652891 frame:
SN: 0013a2004089bfbf
Campos do frame(lumi/temp/humi): 0/600/713
2015-06-15 20:27:03.961281 insertlocaldata: conexao com banco local realizada
2015-06-15 20:27:04.891714 insertlocaldata: Save data from local database
2015-06-15 20:27:04.919363 frame:
SN: 0013a2004089bfbf
Campos do frame(lumi/temp/humi): 0/600/713
2015-06-15 20:27:05.471510 insertlocaldata: conexao com banco local realizada
2015-06-15 20:27:05.566674 insertlocaldata: Save data from local database
2015-06-15 20:27:05.605969 frame:
SN: 0013a2004089bfbf
Campos do frame(lumi/temp/humi): 0/600/713
2015-06-15 20:27:06.031630 insertlocaldata: conexao com banco local realizada
2015-06-15 20:27:06.135754 insertlocaldata: Save data from local database
```

Fonte: Própria autora (2015).

A cada 10 minutos foi executado o script sendtoserver.py que coleta os dados armazenados no banco dados local e os envia ao servidor remoto, gerando o log conforme a Figura 63.

Figura 63 - Log do envio dos dados para o servidor

```
2015-06-14 12:30:43.371840 getlocaldata: conexao com banco local realizada
2015-06-14 12:30:43.650640 getlocaldata: Loaded data from local database
2015-06-14 12:30:43.655909 senddata: Sending samples to the server.
2015-06-14 12:30:46.502282 senddata: All data has been sent to server.
2015-06-14 12:30:46.502568 senddata: Deleting local data.
2015-06-14 12:30:46.547655 senddata: Data has been deleted.
2015-06-14 12:40:42.026483 getlocaldata: conexao com banco local realizada
2015-06-14 12:40:42.056417 getlocaldata: Loaded data from local database
2015-06-14 12:40:42.072207 senddata: Sending samples to the server.
2015-06-14 12:40:43.223411 senddata: All data has been sent to server.
2015-06-14 12:40:43.223698 senddata: Deleting local data.
2015-06-14 12:40:43.245548 senddata: Data has been deleted.
2015-06-14 12:50:42.632997 getlocaldata: conexao com banco local realizada
2015-06-14 12:50:42.662671 getlocaldata: Loaded data from local database
2015-06-14 12:50:42.667845 senddata: Sending samples to the server.
2015-06-14 12:50:44.456117 senddata: All data has been sent to server.
2015-06-14 12:50:44.456402 senddata: Deleting local data.
2015-06-14 12:50:44.478102 senddata: Data has been deleted.
```

Fonte: Própria autora (2015).

Este experimento teve como o objetivo testar a aplicabilidade da RSSF desenvolvida na seção 3.2.3 em um ambiente real, onde os sensores ficaram

exposto às intempéries durante uns dias, enviando dados do ambiente ao coordenador conectado a Raspberry Pi. Na Raspberry Pi estes dados eram armazenados internamente para após um período de tempo enviá-los ao servidor, onde foi criado uma interface gráfica para o usuário acessar os dados. Com a conclusão deste experimento, foi possível verificar a usabilidade dos sensores para a captação de dados durante um período de tempo. Mostrando-se assim como uma solução para o monitoramento das variações climáticas.

5 Conclusões e Trabalhos Futuros

Este trabalho teve como objetivos estudar e avaliar experimentalmente o funcionamento de sensores XBee ZB L/T/H (XS-Z16-CB2R) e os protocolos de roteamento disponíveis, bem como analisar teórica e experimentalmente a viabilidade de um sistema formado por estes nós sensores em uma aplicação caracterizada pela necessidade de captura contínua de dados sobre a umidade, temperatura e luminosidade incidentes no solo de uma área nativa usada para pecuária sob manejo extensivo. A partir da sistematização do conhecimento sobre este problema e sobre redes de sensores sem fio constituídas por nós com protocolo ZigBee, seguida do desenvolvimento de um sistema composto por elementos de hardware e software customizados, foi possível montar um ambiente experimental capaz de promover séries de testes complementares cujos resultados sugerem a viabilidade do uso deste sistema para tratar o problema em questão.

Mais especificamente, por meio do teste de verificação de tensão, foi possível perceber que os sensores operam em uma faixa de 4,5 a 1,8 V. No teste de precisão da temperatura foi possível obter uma equação corrigida para os dados lidos pelos sensores com um erro médio de 0,46 ℃. Além disso, no teste de área de cobertura da RSSF, foi possível verificar o efetivo funcionamento dos sensores na topologia *Mesh*, permitindo a ampliação do alcance da rede. Por fim, foi descrita a aplicação cliente-servidor desenvolvida para a coleta e transmissão de dados com base em sockets TCP, desenvolvida na Linguagem Python, a qual se mostrou efetiva para o agrupamento e transmissão dos dados para um servidor remoto responsável pelo armazenamento dos dados, bem como os dados finais após a implementação física e operacional da RSSF, mostrando-se assim como uma solução para o monitoramento das variações climáticas.

Ao finalizar o desenvolvimento deste trabalho foi possível construir um protótipo funcional da RSSF, onde foi aplicado o conhecimento adquirido. Pois foi possível verificar o funcionamento da RSSF com protocolo ZigBee, onde os sensores XBee enviam o pacote de dados ao coordenador conectado na Raspberry Pi, que armazena os dados em um banco de dados local e periodicamente os envia ao servidor, onde através de uma interface HTML é possível acessá-los.

Devido à dificuldade em encontrar bibliografia para o desenvolvimento da

simulação da RSSF em programas de simulação, não foi possível verificar o impacto do uso dos Kits sensores XBee em diferentes escalas (dezenas ou centenas de sensores em áreas com dezenas ou centenas de hectares), também devido ao trabalho ser desenvolvido em duas cidades distintas (Bagé/RS e Campinas/SP) e ocorrer um problema com a Raspberry Pi que estava em Bagé, não foi possível colocar a RSSF em campos sulinos e medir as microvariações do referido bioma, pois como apresentado na seção 4.1.4, os testes práticos foram realizados em Campinas.

Por fim por falta de tempo para a implementação, foi possível elencar como trabalhos futuros:

a) adaptar a Raspberry Pi para ser alimentada através de baterias;
b) adaptar a Raspberry Pi para conectar através da rede 3G;
c) criar gráficos na interface HTML; e, finalmente
d) a simulação da rede em diferentes escalas.

REFERÊNCIAS

ARDUINO; **What is Arduino?.** Disponível em <http://www.arduino.cc/en/Guide/Introduction>. Acessado em mai. 2015.

BARONTI, Paolo et al. (2007**). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards.** Computer Communications (30). p. 1655-1695. doi:10.1016/j.comcom.2006.12.020.

BEAGLEBOARD. **BeagleBone.** [S.I.], 2012. Disponível em: <http://beagleboard.org/static/bonescript/bone101/index.html#(1)>. Acessado em mai. 2015.

BOECHAT, Vitor Sepulveda; **Redes ZigBee e programas Java aplicados à padronização e melhoria da qualidade dos testes de automação em gasodutos.** UFRJ, 2010. Disponível em <http://monografias.poli.ufrj.br/monografias/monopoli10002125.pdf>. Acessado em 29 de jul. 2014.

BURATTI, Chiara et al. **An Overview on Wireless Sensor Networks Technology and Evolutions.** Sensors (Basel), 2009; 9 (9): 6869-6896; doi: 10.3390/s90906869. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3290495/>. Acesso em: 22 jun. 2014.

CHIP, FTDI. **Virtual COM Port Drives.** Disponível em <http://www.ftdichip.com/Drivers/VCP.htm>. Acessado em jul. 2014.

COMOHACER.EU. **Análisis comparativo de las placas Arduino (oficiales y compatibles)**. julho 2014. Disponível em <http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/>. Acessado em mai. 2015.

CRYOTRONICS, Lake Shore. **Model 455 DSP Gaussmeter.** Disponível em <http://www.lakeshore.com/products/gaussmeters/model-455-dsp-gaussmeter/Pages/Overview.aspx>. Acessado em jul. 2014.

DENNIS, Andrew K. **Raspberry Pi Home Automation with Arduino.** Birmingham: Packt Publishing Limited, 2013.

DIGI, International; **XBee drop-in-networking acessories user's guide.** Disponível em: <http://ftp1.digi.com/support/documentation/90000891_H.pdf>. Acessado em:

jul. 2014.

_____. **Download X-CTU.** Disponível em: <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/X-Ctu>. Acessado em mai. 2014.

_____. **Guia de Programação em Python Digi.** Disponível em: < http://www.digi.com/wiki/developer/index.php/Digi_Python_Programmer%27s_Guide >. Acessado em jul. 2014.

_____. **The Major Differences in the XBee Series 1 vs. the XBee Series 2.** Disponível em: <http://www.digi.com/support/kbase/kbaseresultdetl?id=2213>. Acessado em 15 de ago. 2014.

_____. **XBee End Points.** Disponível em: <http://www.digi.com/wiki/developer/index.php/Understanding_XBee_EndPoints>. Acessado em 14 de ago. 2014.

_____. **XBee Sensors. 2013.** Disponível em: <http://www.digi.com/pdf/ds_xbeesensors.pdf>. Acessado em jun. 2014.

_____. **XBee Sensors.** Disponível em: <http://www.digi.com/wiki/developer/index.php/XBee_Sensors#iDigi_Dia_Configuration_and_Programming_Examples>. Acessado em jun. 2014.

_____. **XBee®/XBee-PRO® ZB RF Modules. 2014.** Disponível em <http://ftp1.digi.com/support/documentation/90000976_S.pdf>. Acessado em jun. 2014.

DEANGELO, Moises Diego; DOMINGUES, Patrícia Silva; PINHO, Leonardo Bidese de. **Implementação e Avaliação de Firmware para Formação de Redes de Sensores sem Fio Aplicadas à Pecuária de Precisão.** Anais do Salão Internacional de Ensino, Pesquisa e Extensão, v. 5, n. 2 (2013). Disponível em: < http://publicase.unipampa.edu.br/index.php/siepe/issue/view/52>. Acessado em jun. 2014.

DOMINGUES, Patrícia Silva; DEANGELO, Moises Diego; PINHO, Leonardo Bidese de. **Soluções para Pecuária de Precisão Baseadas em Redes de Sensores sem Fio.** Anais do Salão Internacional de Ensino, Pesquisa e Extensão, v. 4, n. 2 (2012). Disponível em: <http://publicase.unipampa.edu.br/index.php/siepe/issue/view/40>. Acessado em jun. 2014.

DRUMMOND, José A., FRANCO, José L. A., OLIVEIRA, Daniela. Uma análise sobre a história e a situação das unidades de conservação no Brasil. In: GANEN, Roseli Senna (Org.). **Conservação da biodiversidade: legislação e políticas.** Brasília: Câmara dos Deputados, Edições Câmara, 2010. p. 341-386. Disponível em: <http://ibnbio.org/wp-content/uploads/2012/09/conservacao_biodiversidade1.pdf>. Acessado em: 22 jan. 2014.

ELMOONY. **Schematic-Skema-Raspberry-Pi.** [S.I.], 2012. Disponível em: <https://elmoony.wordpress.com/2012/12/06/perbedaan-raspberry-pi/schematic-skema-raspberry-pi/>. Acessado em mai. 2015.

GISLASON, Drew**. ZigBee Wireless Networking.** Newnes: Elsevier, 2008.

INSTRUTHERM; **Fonte de Alimentação Digital Assimétrica.** Disponível em<http://www.instrutherm.com.br/instrutherm/product.asp?template_id=60&old_template_id=60&partner_id=&tu=b2c&dept%5Fid=2140&pf%5Fid=03281&nome=Fonte+de+Alimenta%E7%E3o+Digital+Assim%E9trica&dept%5Fname=Fontes+de+Alimenta%E7%E3o>. Acessado em 23 de jun. 2014.

KUROSE, James; ROSS, Keith. **Redes de Computadores e a Internet: uma abordagem top-dono.** 3. ed. São Paulo: Pearson Edison Wesley, 2006.

LYNN, August Linse; **Power a Digi LTH Sensor with 5vdc.** 2012. Disponível em <http://lynndia.blogspot.com.br/2012/05/power-digi-lth-sensor-with-5vdc.html>. Acessado em jul. 2014.

MARFIEVICI, R.; et al. **How Environmental Factors Impact Outdoor Wireless Sensor Networks: A Case Study.** Tenth International Conference on Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE. p. 565 – 573. doi: 10.1109/MASS.2013.13. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6680299&tag=1>. Acessado em jun. 2015.

MESSIAS, Antonio Rogério; **Controle remoto e aquisição de dados via XBee/ZigBee (IEEE 802.15.4).** Disponível em: <http://www.rogercom.com/ZigBee/ZigBee.htm>. Acessado em: jul. 2014.

MMA, Ministério do Meio Ambiente. **Bioma Pampa.** [Sol.], 2014. Disponível em: <http://www.mma.gov.br/biomas/pampa>. Acesso em: 22 jan. 2014.

PILLAR, Valério De Pata et al. (Org.). **Campos Sulinos - conservação e uso**

sustentável da biodiversidade. Brasília: MMA, 2009. Disponível em: <http://ecoqua.ecologia.ufrgs.br/arquivos/Livros/CamposSulinos.pdf>. Acesso em: 22 jan. 2014.

PINHEIRO, José Mauricio Santos; **As Redes com ZigBee. Projeto de Redes**, 2004. <http://www.projetoderedes.com.br/artigos/artigo_zigbee.php>. Acessado em 27 de jun. 2014.

QUADROS, Fernando L. F., TRINDADE, José Pedro P., BORBA, Marcos. **A abordagem funcional da ecologia campestre como instrumento de pesquisa e apropriação do conhecimento pelos produtores rurais.** In: PILLAR, Valério De Pata et al. (Org.). Campos Sulinos - conservação e uso sustentável da biodiversidade. Brasília: MMA, 2009. p. 206-213.

RANTS & RAVES-The Blog!; **NOOBS for Raspberry Pi**, junho 2013. Disponível em: <http://qdosmsq.dunbar-it.co.uk/blog/2013/06/noobs-for-raspberry-pi/>. Acessado em abr. 2015.

RASPBERRY PG; **Step 5: installing PostgreSQL on my Raspberry PI,** novembro 2013. Disponível em <http://raspberrypg.org/2013/11/step-5-installing-postgresql-on-my-raspberry-pi/>. Acessado em mai. 2015.

RASPEBERRYPI.ORG; **Installing python packages.** Disponível em <https://www.raspberrypi.org/documentation/linux/software/python.md>. Acessado em abr. 2015.

_____. **NOOBS SETUP.** Disponível em <https://www.raspberrypi.org/help/noobs-setup/>. Acessado em abr. 2015.

RINGWALD, Matthias; ROMER, Kay; **Deployment of Sensor Networks: Problems and Passive Inspection.** Fifth Workshop on Intelligent Solutions in Embedded Systems, 2007. p. 179-192. doi: 10.1109/WISES.2007.4408504. Disponível em < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4408504&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4408504>. Acessado em ago. 2014.

RIVERO, Ilo Amy Saldanha; **Rede de Sensores Sem Fio para monitoramento de equipamentos eletrônicos.** PUC Minhas, 2011. Disponível em: <http://www.biblioteca.pucminas.br/teses/Informatica_RiveroIAS_1.pdf>. Acessado em 22 de mai. 2014.

ROCHA, João Wilson Vieira; **Rede de Sensores Sem Fio: Aplicações.** [Sol.], 2014. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialrssf/pagina_3.asp>. Acesso em: 3 fev. 2014.

SOARES, Sérgio Aurélio Ferreira; **Rede de Sensores Sem Fio para a localização e Monitoramento de pequenos Ruminantes.** UNIVASF, 2012. Disponível em: <http://www.univasf.edu.br/~ccomp/monografias/monografia_8.pdf>. Acessado em: 20 de mai. 2014.

SOUZA, Marco Aurélio; **Sistema de Automação Residencial para Iluminação.** UniCEUB, 2010. Disponível em <http://www.repositorio.uniceub.br/bitstream/123456789/3388/3/20615010.pdf>. Acessado em: jul. 2014.

TECHNOLOGIES, X-IO. **Digi XStick Datasheet.** Disponível em <http://www.x-io.co.uk/downloads/Digi-XStick-Datasheet.pdf>. Acessado em 23 de jun. 2014.

TECNOLOGIES, Keysight; **A Multímetro digital de 6 ½ dígitos.** Disponível em<http://www.home.agilent.com/en/pd-1000001295%3Aepsg%3Apro-pn-34401A/digital-multimeter-6-digit?&cc=BR&lc=por>. Acessado 23 de jun. 2014.

UNIPAMPA. Sistema de Bibliotecas. **Manual para elaboração e normalização de trabalhos acadêmicos – conforme normas da ABNT** / organização Cátia Rosana L. de Araújo, Cristiane Pereira Maciel e Dilva Carvalho Marques. Universidade Federal do Pampa, Sistema de Bibliotecas. Bagé – RS, 2010.