JACINTO CARLOS ASCENCIO CANSADO

Instrumentalização de Redes de Sensores Sem Fio

São Paulo

2011

JACINTO CARLOS ASCENCIO CANSADO

Instrumentalização de Redes de Sensores Sem Fio

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

São Paulo

2011

JACINTO CARLOS ASCENCIO CANSADO

Instrumentalização de Redes de Sensores Sem Fio

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

Área de Concentração: Sistemas Digitais

Orientador:

Prof. Dr. Jorge Rady de Almeida Junior

São Paulo

2011

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de dezembro de 2011.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Cansado, Jacinto Carlos Ascencio
Instrumentalização de Redes de Sensores Sem Fio / J.C.A. Cansado. – ed.rev. – São Paulo, 2011.
181 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Instrumentalização de software 2. Redes de sensores sem fio 3. Restrições temporais 4. Manutenção da qualidade de conexão I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

Dedico este trabalho à minha família. A minha esposa Margarete, que com dedicação e paciência me auxiliou com inúmeras revisões e valiosas sugestões. As minhas filhas Beatriz e Júlia as quais privei de parte da minha dedicação na elaboração desta tese. Aos meus pais, Jacinto e Mara, que embora não estejam mais em nossa companhia, estarão sempre presentes nas minhas lembranças e ações.

AGRADECIMENTOS

Ao professor Dr. Jorge Rady de Almeida Junior, meu orientador, por me guiar e apoiar num dos momentos mais difíceis de minha carreira, pela oportunidade e pela liberdade de realizar este trabalho.

Ao Grupo de Análise de Segurança (GAS) da Escola Politécnica da Universidade de São Paulo (EPUSP), pela oportunidade de realizar o curso de doutorado.

Aos professores e amigos, que de alguma forma contribuíram para a realização deste trabalho.

RESUMO

Este trabalho apresenta um estudo da instrumentalização de software em Redes de Sensores Sem Fio, complementado pela aplicação, em um estudo de caso, de uma instrumentalização. A proposta da instrumentalização visa ampliar o conhecimento sobre essas redes em termos de desempenho, permitindo ao usuário tomar decisões suportadas pelo instrumento. Nesse sentido, foi realizada a abordagem conceitual em termos de Redes de Sensores Sem Fio, da tendência de padronização para essas redes e de instrumentalização por meio de software. Para a comprovação dos benefícios da adoção da utilização do conceito de instrumentalização de software nessas redes, foi desenvolvida uma rede de sensores utilizando esses conceitos, na qual foi possível avaliar o perfil dos aspectos temporais e da manutenção da qualidade da conexão. Os resultados obtidos evidenciam que é possível conhecer, com maiores detalhes, o comportamento da rede, o que melhora a qualidade da tomada de decisão do usuário em relação à sua utilização em aplicações com restrições temporais, nas quais os tempos de resposta envolvidos devem respeitar os limites máximos permitidos.

Palavras-Chave: Instrumentalização de Software. Redes de Sensores Sem Fio. Restrições temporais. Manutenção da qualidade da conexão.

ABSTRACT

This thesis aims at the study of software instrumentation handling in terms of wireless sensor networks which is supplemented with the its application in a case study. The instrumentation purposes go towards the networks knowledge extension concerning its performance by allowing the user to make decisions based on software instrument reading. In this sense, a conceptual approach was performed which takes the wireless sensor network into account as well as the trend of standardization for these networks and the software instrumentation itself. In order to confirm the benefits on adopting the usage of software instrumentation concept on these networks, a wireless sensor network was developed using these concepts. In this sense, it was possible to evaluate the timing aspects profile and the link quality maintenance. The results show that is possible to know about the behavior of the network in details. By doing this, the quality of the user's decision-making regarding their use in applications with time constraints, in which response time involved should respect the maximum allowable limits, has been improved.

Keywords: Software instrumentation. Wireless Sensor Networks. Time constraints. Link quality maintenance.

LISTA DE ILUSTRAÇÕES

FIGURA 1 – ELEMENTOS CONSTITUINTES DE UMA REDE DE SENSORES SEM FIO 24
FIGURA 2 – COMPONENTES DE UM NÓ SENSOR 26
FIGURA 3 – COMPONENTES TÍPICOS DO SUBSISTEMA DE ENERGIA DE UM NÓ SENSOR 32
FIGURA 4 – MODELO DE REFERÊNCIA OSI 47
FIGURA 5 – CAMADAS DE PROTOCOLOS DE REDES DE SENSORES SEM FIO 51
FIGURA 6 – FORMATO DO PACOTE PPDU 55
FIGURA 7 – TOPOLOGIAS ESTRELA E PONTO-A-PONTO 56
FIGURA 8 – ESTRUTURA DO QUADRO DE BEACON – IEEE 802.15.4™-2006 LR-WPAN 58
FIGURA 9 – ESTRUTURA DO QUADRO DE DADOS – IEEE 802.15.4™-2006 LR-WPAN 58
FIGURA 10 – ESTRUTURA DO QUADRO DE CONFIRMAÇÃO – IEEE 802.15.4™-2006 LR-WPAN 59
FIGURA 11 – ESTRUTURA DO QUADRO DE COMANDOS – IEEE 802.15.4™-2006 LR-WPAN 59
FIGURA 12 – ESTRUTURA DO SUPERQUADRO IEEE 802.15.4™-2006 LR-WPAN 61
FIGURA 13 – CAMADAS DE PROTOCOLOS ALIANÇA ZIGBEE™ 64
FIGURA 14 – TOPOLOGIAS DE REDE SUPORTADAS PELO PADRÃO ZIGBEE™ 65
FIGURA 15 – ESTRUTURA DA FERRAMENTA ATOM – FONTE: (SRIVASTAVA; EUSTACE, 1994) 74
FIGURA 16 – POSSÍVEIS MÉTODOS DE INSTRUMENTALIZAÇÃO DE UMA REDE DE SENSORES SEM FIO 80
FIGURA 17 – COMPONENTES TÍPICOS DE UMA REDE DE SENSORES SEM FIO INSTRUMENTALIZADA 86
FIGURA 18 – ESTRUTURA DO PACOTE SONDA 87
FIGURA 19 – ESTRUTURA DA CARGA ÚTIL DO QUADRO DE DADOS SONDA DO NÓ SENSOR 91
FIGURA 20 – ESTRUTURA DA CARGA ÚTIL DO QUADRO DE DADOS SONDA DO NÓ SORVEDOURO 92
FIGURA 21 – FLUXOGRAMA DE INSTRUMENTALIZAÇÃO DO NÓ SENSOR 94
FIGURA 22 – FLUXOGRAMA DE INSTRUMENTALIZAÇÃO DO NÓ SORVEDOURO 96
FIGURA 23 – REGISTRO ENVIADO PELO NÓ SORVEDOURO AO INSTRUMENTO DE SOFTWARE 97
FIGURA 24 – ESTRUTURA DO INSTRUMENTO DE SOFTWARE 99
FIGURA 25 – ESTRUTURA DO REGISTRO ARMAZENADO PELO INSTRUMENTO DE SOFTWARE 100
FIGURA 26 – DIAGRAMA DE RELACIONAMENTO DOS ASPECTOS TEMPORAIS DA REDE 102
FIGURA 27 – DIAGRAMA DE RELACIONAMENTO DA MANUTENÇÃO DA QUALIDADE DA CONEXÃO 103
FIGURA 28 – ARQUITETURA DO AMBIENTE DE TESTES 104
FIGURA 29 - ARQUITETURA DE SOFTWARE DA INSTRUMENTALIZAÇÃO DE REDES DE SENSORES SEM FIO ... 106
FIGURA 30 – IHM – MÓDULO INTERFACE COM O USUÁRIO - CONFIGURAÇÃO 107
FIGURA 31 – MÓDULO INTERFACE COM O USUÁRIO – ASPECTOS TEMPORAIS 108
FIGURA 32 – MÓDULO INTERFACE COM O USUÁRIO – MANUTENÇÃO DA QUALIDADE DA CONEXÃO 109
FIGURA 33 – PSEUDOCÓDIGO DO MÓDULO DE ARMAZENAMENTO DE DADOS 110
FIGURA 34 – PSEUDOCÓDIGO DO MÓDULO DE COMUNICAÇÃO 111
FIGURA 35 – NÓ SENSOR BASEADO NO CIRCUITO INTEGRADO ATMEGA128RFA1 113
FIGURA 36 – PLACA INTERLIGAÇÃO NÓ SENSOR 114
FIGURA 37 – AMBIENTE DE DESENVOLVIMENTO 115

FIGURA 38 – PSEUDOCÓDIGO DO PROGRAMA "SORVEDOURO.C" 118
FIGURA 39 – PSEUDOCÓDIGO DO PROGRAMA “SENSOR.C” 118
FIGURA 40 – VISÃO SISTÊMICA DOS COMPONENTES DE HARDWARE E SOFTWARE UTILIZADOS NA CONSTRUÇÃO DO AMBIENTE DE TESTES 119
FIGURA 41 – CANAL SELECIONADO DO PADRÃO IEEE-802.15.4™-2006-LR-WPAN 120
FIGURA 42 – PLANTA BAIXA DO AMBIENTE FÍSICO UTILIZADO NOS TESTES 123
FIGURA 43 – VALORES DO TEMPO DE IDA-E-VOLTA OBTIDOS DE FORMA INDIVIDUAL 128
FIGURA 44 – VALORES DO TEMPO DE IDA-E-VOLTA OBTIDOS DE FORMA COLETIVA 129
FIGURA 45 – DISTRIBUIÇÃO DE FREQUÊNCIA DO TEMPO DE IDA-E-VOLTA DO NÓ SENSOR 1 OPERANDO DE FORMA INDIVIDUAL 130
FIGURA 46 – DIAGRAMA DE BARRAS COMPARATIVO DO NÓ SENSOR 1 OPERANDO DE FORMA INDIVIDUAL E COLETIVA 130
FIGURA 47 – VALORES DA ENERGIA DETECTADA (ED) OBTIDOS DE FORMA INDIVIDUAL E COLETIVA 135
FIGURA 48 – VALORES DO INDICADOR DA QUALIDADE DA CONEXÃO (LQI) OBTIDOS DE FORMA INDIVIDUAL E COLETIVA 136
FIGURA 49 – VALORES DA TAXA DE RECEPÇÃO DE PACOTES (PRR) OBTIDOS DE FORMA INDIVIDUAL E COLETIVA 137
FIGURA 50 – VALORES DA TENSÃO DA BATERIA (TB) OBTIDOS DE FORMA INDIVIDUAL E COLETIVA 138
FIGURA 51 – VALORES DA TEMPERATURA DE OPERAÇÃO (TO) OBTIDOS DE FORMA INDIVIDUAL E COLETIVA 139

LISTA DE QUADROS

QUADRO 1 – PRINCIPAIS CARACTERÍSTICAS DOS NÓS SENSORES COMERCIAIS – PARTE (A) 28
QUADRO 2 – PRINCIPAIS CARACTERÍSTICAS DOS NÓS SENSORES COMERCIAIS – PARTE (B) 29
QUADRO 3 – PRINCIPAIS CARACTERÍSTICAS DA CAMADA FÍSICA DO PADRÃO LR-WPAN 52
QUADRO 4 – NÍVEIS DE SEGURANÇA DISPONÍVEIS NO PADRÃO ZIGBEE™ 68
QUADRO 5 – RESUMO DAS CARACTERÍSTICAS DOS MECANISMOS DE INSTRUMENTALIZAÇÃO 81
QUADRO 6 – MÉTRICAS INSTRUMENTALIZADAS 88
QUADRO 7 – RESUMO DAS INFORMAÇÕES DOS PACOTES SONDA 93
QUADRO 8 – CARACTERÍSTICAS CONFIGURADAS DA PORTA DE COMUNICAÇÃO SERIAL 110
QUADRO 9 – CARACTERÍSTICAS DA REDE DE SENSORES SEM FIO UTILIZADA NO AMBIENTE DE TESTES 121
QUADRO 10 – CONFIGURAÇÕES UTILIZADAS NO LEVANTAMENTO DO PERFIL DOS ASPECTOS TEMPORAIS 126
QUADRO 11– VALORES OBTIDOS NO LEVANTAMENTO DO PERFIL DOS ASPECTOS TEMPORAIS DE FORMA INDIVIDUAL 127
QUADRO 12 – VALORES OBTIDOS NO LEVANTAMENTO DO PERFIL DOS ASPECTOS TEMPORAIS DE FORMA COLETIVA 127
QUADRO 13 – CONFIGURAÇÕES UTILIZADAS NO LEVANTAMENTO DO PERFIL DA MANUTENÇÃO DA QUALIDADE DA CONEXÃO 132
QUADRO 14 – VALORES OBTIDOS NO LEVANTAMENTO DO PERFIL DA QUALIDADE DA CONEXÃO DE FORMA INDIVIDUAL 133
QUADRO 15 – VALORES OBTIDOS NO LEVANTAMENTO DO PERFIL COLETIVO DOS ASPECTOS DA MANUTENÇÃO DA QUALIDADE DA CONEXÃO 134

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
AODV	Ad-hoc On-demand Distance Vector
API	Application Program Interface
APS	Application Support sub-layer
ASCII	American Standard Code for Information Interchange
ASK	Amplitude Shift Keying
ATOM	Analysis Tool with OM
BER	Bit Error Rate
BI	Beacon Interval
BIT	BInary digiT
BO	Beacon Order
BPSK	Binary Phase-Shift Keying
CAP	Contention Access Period
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCA	Clear Channel Assessment
CCM	Counter with CBC-MAC
CD-ROM	Compact Disk – Read Only Memory
CFP	Contention Free Period
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRI	Criptografado
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
CSV	Comma Separated Values
CTR	Counter

DEC	Digital Equipment Corporation
DSSS	Direct Sequence Spread Spectrum
ED	Energy Detection
EEPROM	Electrically-Erasable Programmable Read-Only Memory
FCS	Frame Check Sequence
FFD	Full-Function Device
GPS	Global Positioning System
GTS	Guaranteed Time Slot
GUI	Graphical User Interface
HTTP	Hiper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
JTAG	Join Test Action Group
Kbps	Kilo bit per second
LED	Light Emitting Diode
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Networks
MAC	Media Access Control
MANET	Mobile Ad hoc NETwork
MEMS	Micro-ElectroMechanical System
MFR	MAC Footer
MHR	MAC Header
MIC	Message Integrity Code
MPDU	MAC protocol data unit
NI	National Instruments

OM	Object Module
O-QPSK	Offset Quadrature Phase-Shift Keying
OSI	Open Systems Interconnection
PAN	Personal Area Network
PHR	PHY header
PPDU	PHY Protocol Data Unit
PRR	Packet Reception Rate
PSDU	PHY Service Data Unit
PWM	Pulse Width Modulation
QoS	Quality of Service
RCBBB	Radio Control Board Breakout Board
RFD	Reduced-Function Device
RISC	Reduced Instruction Set Computing
RSSF	Rede de Sensores Sem Fio
RTOS	Real-Time Operating System
RTT	Round-Trip Time
SD	Superframe Duration
SFD	Start-of-Frame Delimiter
SHR	Synchronization Header
SO	Superframe Order
SRAM	Static Random Access Memory
TB	Tensão da Bateria
TCP/IP	Transmission Control Protocol/Internet Protocol
TO	Temperatura de Operação
USA	United States of America
USART	Universal Synchronous and Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
WSAN	Wireless Sensor and Actuator Network
WWW	World Wide Web
ZDO	ZigBee Device Object

SUMÁRIO

1 INTRODUÇÃO 19

1.1 JUSTIFICATIVA 21

1.2 OBJETIVOS 21

1.3 ESTRUTURA DO TRABALHO 22

2 REDE DE SENSORES SEM FIO 24

2.1 ESTRUTURA DE REDES DE SENSORES SEM FIO 25

2.1.1 Unidade de sensoriamento 27

2.1.2 Unidade de processamento 27

2.1.3 Unidade de comunicação 29

2.1.4 Unidade de energia 31

2.1.5 Sistema de localização 32

2.1.6 Atuadores 33

2.2 APLICAÇÕES DE REDES DE SENSORES SEM FIO 34

2.2.1 Características das Redes de Sensores Sem Fio 34

2.2.1.1 Aplicação específica 35

2.2.1.2 Interação com o meio físico 35

2.2.1.3 Escalabilidade 35

2.2.1.4 Energia 36

2.2.1.5 Auto-organização 36

2.2.1.6 Dependabilidade e qualidade de serviço 36

2.2.1.7 Arquitetura de redes centrada em dados 37

2.2.1.8 Simplicidade 37

2.2.2 Classificação das Redes de Sensores Sem Fio 37

2.2.2.1 Detecção de eventos e transmissão da informação 38

2.2.2.2 Aquisição e transmissão da informação de forma periódica 39

2.2.2.3 Consulta de informações pela aplicação 40

2.2.2.4 Rastreamento 41

2.3 DESAFIOS EM REDES DE SENSORES SEM FIO 41

2.4 MANUTENÇÃO DA QUALIDADE DA CONEXÃO EM REDES DE SENSORES SEM FIO 45

3 PADRÕES PARA REDES DE SENSORES SEM FIO 47

3.1 MODELO DE REFERÊNCIA OSI 47

3.2 PADRÃO IEEE STD 802.15.4™-2006 LR-WPAN 51

3.2.1 Camada física 52

3.2.2 Camada controle de acesso ao meio 55

3.2.2.1 Quadro de beacon 57

3.2.2.2 Quadro de dados 58

3.2.2.3 Quadro de confirmação 59
3.2.2.4 Quadro de comando 59
3.3 PADRÃO ZIGBEE™ 63
3.3.1 Topologia de rede 64
3.3.2 Roteamentos suportados 65
3.3.3 Suporte à segurança de dados 67
3.3.4 Entrada e saída de dispositivos da rede 69
4 INSTRUMENTALIZAÇÃO DE SOFTWARE 71
4.1 CONCEITOS DE INSTRUMENTALIZAÇÃO DE SOFTWARE 71
4.2 PRINCIPAIS FORMAS DE IMPLANTAÇÃO 75
4.2.1 Métodos de Instrumentalização de Software em Redes de Sensores Sem Fio 78
4.3 APLICAÇÕES 81
5 PROPOSTA DE INSTRUMENTALIZAÇÃO PARA REDES DE SENSORES SEM FIO 84
5.1 CONSIDERAÇÕES SOBRE A PROPOSTA E CONTRIBUIÇÕES PRINCIPAIS 84
5.2 INSTRUMENTALIZAÇÃO DE REDES DE SENSORES SEM FIO SUPORTADA PELO PADRÃO IEEE 802.15.4™ – 2006 LR-WPAN 85
5.2.1 Instrumentalização do nó sensor e do nó sorvedouro 86
5.2.2 Instrumento de Software 98
5.3 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO 103
6 AMBIENTE DE TESTES 104
6.1 ARQUITETURA DO AMBIENTE DE TESTES 104
6.2 INSTRUMENTO DE SOFTWARE 105
6.2.1 Hardware do Instrumento de Software 106
6.2.2 IHM - Interface Homem Máquina 106
6.2.3 Armazenamento de dados 109
6.2.4 Comunicação serial com o nó sorvedouro 110
6.3 REDE DE SENSORES SEM FIO INSTRUMENTALIZADA 111
6.3.1 Hardware do nó sensor 111
6.3.2 Hardware do nó sorvedouro 114
6.3.3 Ambiente de desenvolvimento de software 115
6.3.4 Instrumentalização dos nós sensores e do nó sorvedouro 116
6.4 CARACTERÍSTICAS DA REDE DE SENSORES SEM FIO UTILIZADA NO AMBIENTE DE TESTES 120
6.5 AMBIENTE DE TESTES 122
6.5.1 Ambiente físico da realização dos testes 122
6.6 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO 123
7 RESULTADOS DA INSTRUMENTALIZAÇÃO 124
7.1 RESULTADOS DA INSTRUMENTALIZAÇÃO NO LEVANTAMENTO DO PERFIL DOS ASPECTOS TEMPORAIS 124

7.1.1 Instrumentalização dos Aspectos Temporais 126
7.1.1.1 Instrumentalização dos Aspectos Temporais Individuais 127
7.1.1.2 Instrumentalização dos Aspectos Temporais Coletivos.....127
7.1.1.3 Análise Gráfica dos Aspectos Temporais 128
7.1.1.4 Comentários sobre o perfil dos Aspectos temporais.....131
7.2 RESULTADOS DA INSTRUMENTALIZAÇÃO NO LEVANTAMENTO DO PERFIL DA MANUTENÇÃO DA QUALIDADE DA CONEXÃO 131
7.2.1 Instrumentalização para Verificação da Qualidade da Conexão 132
7.2.1.1 Instrumentalização para Verificação da Qualidade da Conexão Individual.....133
7.2.1.2 Instrumentalização para Verificação da Qualidade da Conexão Coletiva 134
7.2.1.3 Análise gráfica dos aspectos da manutenção da qualidade da conexão 134
7.2.1.4 Comentários sobre o perfil da manutenção da qualidade da conexão.....140
7.3 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO 140
8 CONSIDERAÇÕES FINAIS 141
8.1 CONTRIBUIÇÕES E CONCLUSÕES..... 141
8.2 TRABALHOS FUTUROS 143
9 REFERÊNCIAS..... 145

1 INTRODUÇÃO

As Redes de Sensores Sem Fio (RSSF) estão mudando a forma com que as pessoas interagem com o mundo físico. Um número considerável de nós sensores, cada um possuindo capacidade limitada de processamento, de sensoriamento, de memória, de autonomia e de comunicação, é distribuído, de forma organizada ou aleatória. Essa distribuição é realizada em uma determinada área para obter dados de interesse do meio físico, com o objetivo de executar o processamento local desses dados e transmitir os resultados para uma entidade central, denominada nó sorvedouro, com maior capacidade computacional, no sentido de disponibilizar essa informação para pessoas interessadas.

As aplicações mais usuais para essas redes são de monitoramento ambiental e rastreamento de algum objeto de interesse. Embora a tecnologia de redes de sensores sem fio seja, na grande maioria das vezes, utilizada em aplicações com baixa taxa de amostragem, baixa largura de banda e baixa complexidade computacional, existem classes de aplicações com características diferentes destas apresentadas. Nesse contexto, pode-se citar como exemplos, a automação industrial, a telemedicina, a automação bancária, a automação comercial, o sistema de detecção de intrusão, o controle de processo, o suporte à vida e a vigilância por meio de imagens, entre outras. Essas aplicações requerem maior capacidade computacional, maior autonomia em termos energéticos e também conformidade e aderência aos requisitos temporais, tal como, o tempo de atraso da disponibilidade da informação, em termos de aquisição, processamento e comunicação.

Em sua maioria, as Redes de Sensores Sem Fio, possuem recursos limitados, baixa confiabilidade em termos de nós individuais, arquitetura distribuída, comunicação sem fio e topologia de rede dinâmica. Essas características proporcionam desafios consideráveis a serem equacionados no desenvolvimento dessas novas classes de aplicações, que são denominadas de “segunda geração de aplicações de RSSF” (PAGANO; CHITINIS; LIPARI, 2007).

Existe um número elevado de estudos sobre essas redes, porém proporcionalmente um número ínfimo de estudos sobre Redes de Sensores Sem Fio envolvendo sistemas com restrições temporais e de natureza de tempo real, caracterizados por serem sistemas reativos aos estímulos do ambiente e com prazos temporais bem definidos. Devido à natureza autônoma dos nós sensores de uma Rede de Sensores Sem Fio, a eficiência energética do sistema como um todo é uma tônica constante presente na maioria dos estudos e das aplicações. Essa característica intrínseca impõe uma situação de compromisso entre consumo de energia e de desempenho em termos de compromissos temporais.

O estudo aprofundado das origens e a mensuração dos atrasos temporais presentes nessas redes é uma tarefa complexa e desafiante. As aplicações com requisitos de tempo real possuem prazos (intervalos de tempo) bem definidos, que devem ser cuidadosamente observados no sentido de não serem ultrapassados, situação que pode levar o sistema a uma condição indesejada. Uma resposta “atrasada” constituí-se em uma falha do sistema, e dependendo da natureza da aplicação, pode causar danos irreparáveis.

Em Redes de Sensores Sem Fio, um atraso pode ser considerado como o intervalo de tempo entre a ocorrência do fenômeno físico de interesse e a disponibilidade da informação para a tomada de decisão. Nesse processo estão relacionados os atrasos no estágio de sensoriamento, no estágio de processamento de diversas atividades pelo processador, utilizando ou não um Sistema Operacional de Tempo Real – *Real-Time Operating System* (RTOS), no estágio de processamento das camadas de comunicação, entre eles nos protocolos de controle de acesso ao meio – *Media Access Control* (MAC), nos protocolos da camada de rede, nos protocolos da camada de transporte, e finalmente, no canal de comunicação. Processo semelhante ocorre no nó receptor.

Nesse sentido, a disponibilidade de um instrumento que possibilite o conhecimento do comportamento da Rede de Sensores Sem Fio em termos dos aspectos temporais e também da manutenção da qualidade da conexão, fornece uma ferramenta importante para a análise dessas redes.

1.1 JUSTIFICATIVA

A necessidade crescente de se construir sistemas computacionais confiáveis, robustos, tolerantes a falhas e previsíveis têm se tornado um desafio cada vez mais difícil de ser alcançado com as tecnologias convencionais, em virtude do aumento da demanda de aplicações complexas que também apresentam requisitos temporais cada vez mais rígidos (LI; THAI; WU, 2008).

O desenvolvimento da microeletrônica proporciona o surgimento de componentes eletrônicos com dimensões reduzidas, menores preços, maior capacidade funcional e menor consumo de energia. Essa tendência possibilita o desenvolvimento de aplicações mais complexas e economicamente viáveis e que era proibitivo anteriormente. Existem grupos de aplicações sensíveis aos quesitos temporais e que podem se beneficiar dessa nova tecnologia de Redes de Sensores Sem Fio, se for utilizada de forma consciente e se forem observadas as limitações inerentes à tecnologia (CULLER; ESTRIN; SRIVASTAVA, 2004).

Não há uma orientação teórica consistente, que ofereça suporte para o desenvolvimento de ferramentas para a avaliação das características de interesse de Redes de Sensores Sem Fio, por exemplo, em relação aos aspectos temporais envolvidos e em termos da manutenção da qualidade da conexão. Essa lacuna deve-se ao fato dessas redes serem uma tecnologia recente e, principalmente, a sua dependência em relação à aplicação, características que dificultam a utilização de ferramentas existentes para redes convencionais e também o desenvolvimento de novas ferramentas que abordem as Redes de Sensores Sem Fio, de forma eficiente (SUNDANI et al., 2011).

1.2 OBJETIVOS

O principal objetivo deste trabalho é a utilização do conceito de instrumentalização de software em uma Rede de Sensores Sem Fio, na qual informações de interesse

são adicionadas aos pacotes de dados que trafegam pela rede e que auxiliam no diagnóstico de seu desempenho. Essa sobrecarga de informação não deve afetar aplicações que possuam restrições temporais.

Mais especificamente a proposta consiste na utilização do conceito da instrumentalização de software em uma Rede de Sensores Sem Fio, por meio do desenvolvimento de um instrumento de software com duas finalidades: interface com o usuário e instrumentalização dos nós constituintes da rede, ou seja, sensores e sorvedouro. A aplicação desse conceito possibilita, por exemplo, realizar a análise dos aspectos temporais e da manutenção da qualidade da conexão, além de outros aspectos de interesse envolvidos na utilização consciente dessa tecnologia, visando sua utilização em aplicações sensíveis ao tempo real.

Outro objetivo é o de implementar e testar a instrumentalização de software em uma rede prática com o intuito de se avaliar as características dessa rede em termos de aspectos temporais e de manutenção da qualidade da conexão, fatores importantes em aplicações sujeitas a restrições temporais.

Esta proposta permite ampliar o conhecimento sobre Redes de Sensores Sem Fio em termos de limitações, padronizações e desafios existentes, possibilitando o desenvolvimento de aplicações utilizando essa tecnologia.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em capítulos, cada qual abordando assuntos necessários ao embasamento da pesquisa. O presente capítulo faz uma introdução ao trabalho, bem como apresenta os objetivos e as justificativas do mesmo.

O capítulo 2 fornece uma visão abrangente e geral da tecnologia de Redes de Sensores Sem Fio, em termos de estrutura, classificação dessas redes em função das aplicações, desafios envolvidos nessa tecnologia e aspectos referentes à manutenção da qualidade da conexão.

O capítulo 3 aborda a questão da tendência de padronização das Redes de Sensores Sem Fio, utilizando como base o modelo de referência – *Open Systems Interconnection* (OSI) e apresentando os dois principais padrões que estão sendo utilizados na tecnologia nesses tipos de rede, ou seja, o IEEE 802.15.4™ LR-WPAN e o ZigBee™.

O capítulo 4 apresenta o conceito de instrumentalização de software, as principais formas de implantação e também as aplicações de Redes de Sensores Sem Fio que implantam parcialmente esse conceito.

O capítulo 5 descreve a proposta da aplicação do conceito de instrumentalização de software para Redes de Sensores Sem Fio, abordando as considerações e contribuições da proposta e sua implantação conceitual em uma rede suportada pelo padrão IEEE 802.15.4™-2006 LR-WPAN, envolvendo todos os elementos constituintes do processo de instrumentalização.

O capítulo 6 descreve o ambiente de testes utilizado, apresentando sua arquitetura, o instrumento de software e a instrumentação dos elementos componentes da rede. Adicionalmente nesse capítulo são descritos as características da Rede de Sensores Sem Fio, como também o ambiente de testes.

O capítulo 7 é reservado para a descrição dos testes e apresentação dos resultados da utilização do conceito de instrumentalização de software em uma Rede de Sensores Sem Fio.

Finalmente, o capítulo 8 é dedicado às considerações finais do desenvolvimento do trabalho relacionando as contribuições e conclusões, bem como a elaboração de propostas para futuros trabalhos na área de instrumentalização de software em Redes de Sensores Sem Fio.

2 REDE DE SENSORES SEM FIO

Os avanços na tecnologia de fabricação dos semicondutores proporcionam o desenvolvimento de dispositivos cada vez menores, com custos mais acessíveis e com capacidades funcionais crescentes. Essas características promovem a criação de componentes “inteligentes”, como sensores e pequenos atuadores. Atualmente esses elementos possuem processamento local e comunicação por rede sem fio (*wireless*). A utilização de dezenas ou até milhares desses dispositivos em uma rede de comunicação auto-configurável originou a tecnologia conhecida como Rede de Sensores sem Fio (RSSF).

Existe um número expressivo de trabalhos sobre pesquisas em Redes de Sensores Sem Fio e todos são unânimes em justificar que o desenvolvimento dessa tecnologia é resultado da inevitável convergência da comunicação sem fio, da computação e dos dispositivos sensores, proporcionada pela tecnologia dos semicondutores (AKYLDIZ et al., 2002; POTTIE, 1998; POTTIE; KAISER, 2000; STANKOVIC, 2008a; STANKOVIC, 2008b; TUBAISHAT; MADRIA, 2003; VIEIRA et al., 2003). A figura 1 ilustra a união dessas funcionalidades na obtenção de um nó sensor e a sua interligação em rede proporcionando a formação de uma Rede de Sensores Sem Fio.

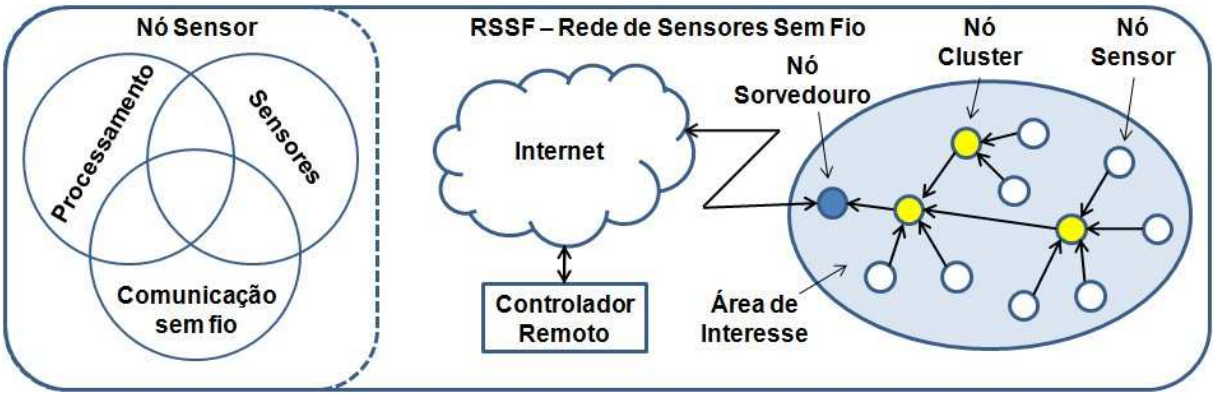


Figura 1 – Elementos constituintes de uma Rede de Sensores Sem Fio

A estrutura dessa rede está baseada na comunicação de nós sensores independentes que incorporam um microprocessador, memórias, sensores, módulo

de rádio e bateria. Devido às características intrínsecas dessa tecnologia, o uso de baterias impõe limitações de armazenamento, de processamento e de comunicação sem fio não estruturada, ou seja, ad hoc, com o agravante, em muitas oportunidades, dos mesmos estarem dispersos no ambiente de maneira aleatória e não assistida. Essas características proporcionam baixa confiabilidade individual aos nós sensores, que por sua vez é compensada pelo trabalho colaborativo da rede e também pela redundância devido à elevada quantidade desses dispositivos.

Os outros elementos constituintes da Rede de Sensores Sem Fio são os nós sorvedouros e os nós clusters. O primeiro elemento é obrigatório e tem como principal função ser o ponto de acesso da rede, responsável pela comunicação com os demais elementos. Já a utilização de nós clusters é dependente da aplicação. Sua função principal é agregar os dados dos nós vizinhos, criando uma hierarquia de comunicação com o nó sorvedouro. Em geral esses componentes possuem disponibilidade de recursos muito superior aos dos nós sensores, por exemplo, em termos de memória, de processamento, de energia e de conectividade, entre outros.

Projetos de sistemas com “inteligência” distribuída, com sensores locais distribuídos no ambiente físico, com capacidade de processamento de uma lógica pré-definida, atuadores, e ainda auxiliados pelo uso da tecnologia sem fio, têm proporcionado uma considerável redução de custos, principalmente em termos de distribuição de cabos, permitindo uma maior flexibilidade na instalação. Conexões sem fio em áreas perigosas possibilitam a redução de riscos pessoais devido ao fato de não haver a necessidade de expor pessoas a essas condições. Outro diferencial dessa tecnologia é a rápida possibilidade de sua utilização em locais sem estrutura física pré-existente (ponto de energia, cabeamento estruturado, etc.), o que é praticamente inviável em termos de tecnologia de redes convencionais a cabo.

2.1 ESTRUTURA DE REDES DE SENSORES SEM FIO

Basicamente uma Rede de Sensores Sem Fio é constituída pelos seguintes elementos: nós sensores, nós clusters e nós sorvedouros. Um nó sensor típico é

composto por quatro componentes: a unidade de sensoriamento, a unidade de processamento, a unidade de comunicação e a unidade de energia. Alguns tipos de aplicações podem necessitar de componentes adicionais, tais como sistema de localização, por exemplo, a utilização de sistema de posicionamento global – *Global Positioning System* (GPS), geradores de energia e ainda atuadores. Esses componentes são interligados conforme a figura 2 (AKYLDIZ et al., 2002). Os componentes adicionais estão destacados na figura em linha tracejada fina. As setas em vermelho indicam o fluxo de energia e as setas em preto sinalizam o fluxo de dados.

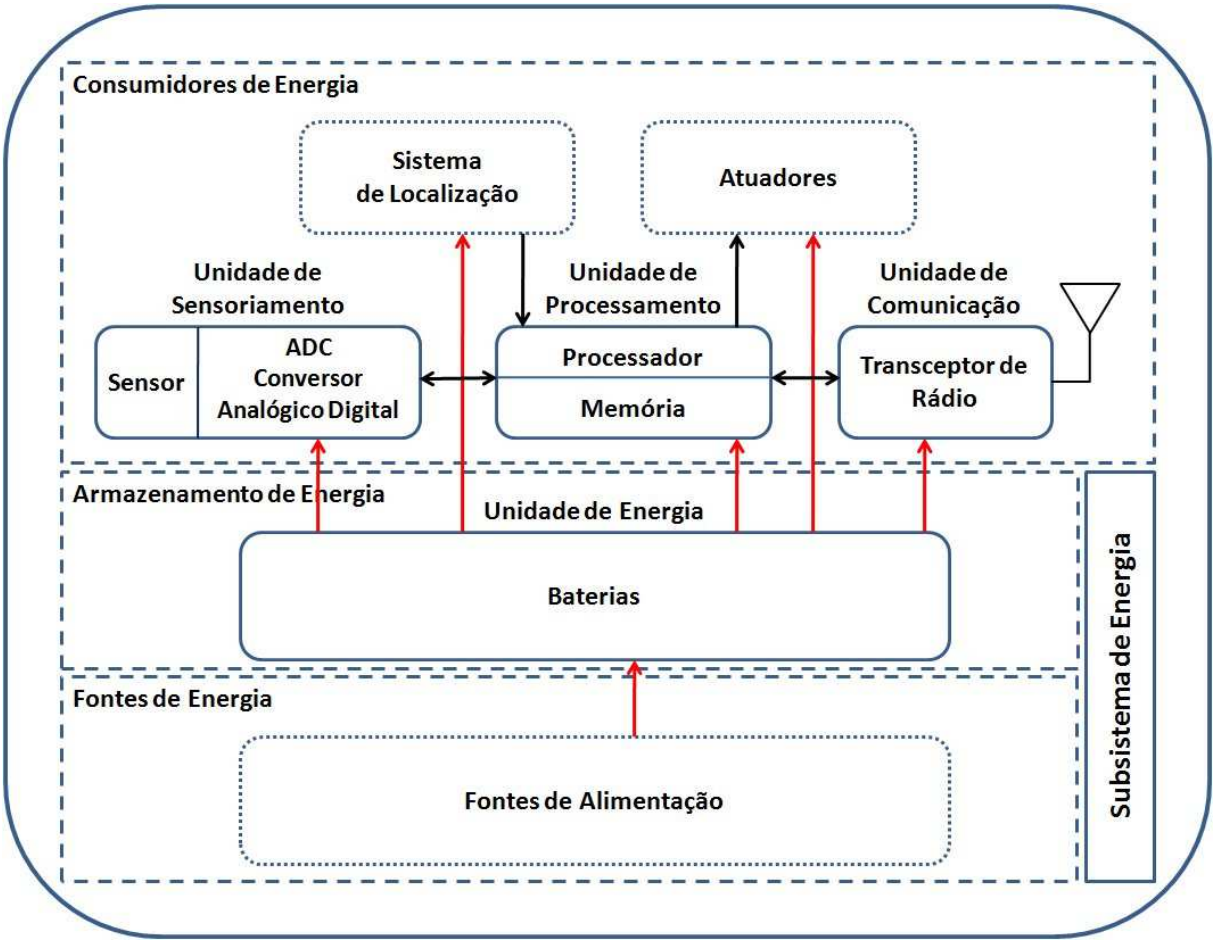


Figura 2 – Componentes de um nó sensor

A seguir são detalhados os componentes de um nó sensor.

2.1.1 Unidade de sensoriamento

Esta unidade constitui os “olhos” e os “ouvidos” do nó sensor. É formada por materiais que apresentam alteração em sua característica elétrica quando sujeitos às variações das condições ambientais. O termistor é um componente eletrônico cuja resistência varia em função da temperatura existente ao seu redor. O mesmo conceito pode ser aplicado aos sensores de estresse (*strain gauge*), cuja resistência elétrica varia em função da força mecânica aplicada ao sensor. Um circuito dedicado faz a conversão da grandeza analógica para digital – *Analog to Digital Converter* (ADC). Esse valor digital pode então ser armazenado e trabalhado pelo processador. Existem vários tipos de sensores para serem utilizados na detecção de diversos fenômenos, tais como: luminosidade, aceleração, impacto, pressão, vazão e umidade, entre outros.

O desenvolvimento de dispositivos baseados em micro sistemas eletromecânicos – *Micro Electro-Mechanical Systems* (MEMS) tem proporcionado a criação de uma gama razoável de componentes para monitoração de vários fenômenos físicos de uma forma eficiente e relativamente barata. Os sensores baseados nessa tecnologia podem monitorar várias forças, concentrações de elementos químicos e fatores ambientais. Enquanto acelerômetros piezoelétricos de alta precisão apresentam um custo da ordem de centenas de dólares, um componente baseado na tecnologia MEMS fornece a precisão necessária ao custo de algumas dezenas de dólares. Outro ponto favorável é em relação à possibilidade da produção em massa, incorporando os benefícios do desenvolvimento dos semicondutores, melhorando a precisão e com custos atrativos (CULLER; ESTRIN; SRIVASTAVA, 2004).

2.1.2 Unidade de processamento

Esta unidade, geralmente associada a um sistema de armazenamento limitado, é responsável pelo gerenciamento das atividades necessárias no sentido de disseminar os dados coletados de forma eficiente, por meio de trabalho colaborativo

com outros nós da rede. Como resultado do desenvolvimento dos circuitos semicondutores, os microcontroladores ficaram menores, levando a um consumo de energia reduzido para uma determinada frequência de operação e utilizando uma área física menor. Existe uma linha de desenvolvimento de microcontroladores que prioriza a eficiência ao invés da adição de novas funcionalidades ao dispositivo, possibilitando o consumo de energia próximo de 1 mW com uma frequência de operação em torno de 10 MHz (CULLER; ESTRIN; SRIVASTAVA, 2004). Os Quadros 1 e 2 a seguir, relacionam as principais características de alguns nós sensores comerciais utilizados nos projetos de Redes de Sensores Sem Fio.

Características	**Nós sensores comerciais**			
	Btnode3	**Mica2**	**Mica2dot**	**MicaZ**
Fabricante	Art Technology	Crossbow		
Microcontrolador	Atmel - Atmega 128L			
Freq.clock (MHz)	7,37		4	7,37
RAM (KB)	64 + 180	4		
ROM (KB)	128			
Storage (KB)	4	512		
Rádio	Chipcom CC1000			CC2420
Taxa (Kbauds)	38,4			250
Alcance máx. (m)	150 ~ 300			75 ~ 100
Bateria	2AA		Moeda	2AA
Interface computador	Placa programação			
Sistema operacional	Nut OS	Tiny OS		
Transdutores	Placa aquisição			
Extras	Bluetooth			

Quadro 1 – Principais características dos nós sensores comerciais – parte (a)

Características	Nós sensores comerciais			
	Telos A	**Tmote Sky**	**Eyes**	**RCB 128RFA1**
Fabricante	Imote IV		Univ. Twente	Dresden
Microcontrolador	Atmel - Atmega 128L			
Freq.clock (MHz)	8		5	16
RAM (KB)	2	20	2	16
ROM (KB)	60	48	60	128
Storage (KB)	256	1024	4	
Rádio	Chipcom CC2420		RFM TR10011868	AT86RF231
Taxa (Kbauds)	250		57,8	250
Alcance máx. (m)	75 ~ 100			
Bateria	2AA			2AAA
Interface computador	USB		Porta serial COM	
Sistema operacional	Tiny OS		PEEROS	
Transdutores	Placa CPU		Placa aquisição	
Extras				RCB Integrado

Quadro 2 – Principais características dos nós sensores comerciais – parte (b)

2.1.3 Unidade de comunicação

É um dos estágios mais desafiadores em termos de complexidade. Boa parte das Redes de Sensores Sem Fio operam na forma multi-saltos (multi-hop) e são suportadas por um modelo de protocolos composto por camadas de comunicação que inclui a camada física, a camada de controle de acesso ao meio, a camada de rede e, finalmente, a camada de transporte. Existem inúmeros protocolos disponíveis para essas camadas, porém eles não são os mesmos protocolos disponíveis para redes sem fio convencionais, como por exemplo, as redes que seguem o padrão internacional elaborado pelo *Institute of Electrical and Electronics Engineers* (IEEE), sob a identificação de IEEE 802.11.

A necessidade por soluções diferenciadas é ocasionada pelas características e restrições dos dispositivos “inteligentes” e do ambiente monitorado. Por exemplo, pequenos rádios não executam detecção de colisão de pacotes, durante a transmissão e recepção dos mesmos, no sentido de melhorar a confiabilidade da

comunicação, tabelas de roteamento complexas não podem ser armazenadas nas memórias escassas dos nós sensores e a quantidade limitada de energia reduz o alcance do link de rádio e muitas vezes força com que os dispositivos fiquem inativos por longos períodos de tempo, no sentido de economizar energia. Adicionalmente, o tráfego padrão de pacotes em uma Rede de Sensores Sem Fio inclui uma coleção de dados que são aquisitados e enviados para a estação base, denominada de nó sorvedouro. Outro padrão frequente é a disseminação de dados de controle ou de requisição para todos os dispositivos da rede. A comunicação arbitrária de dispositivo-para-dispositivo, conhecida como comunicação ponto-a-ponto, embora comum em outros tipos de redes, não é usual em Redes de Sensores Sem Fio. Outro desafio é que o ambiente ruidoso é prejudicial para redes sem fio.

Pelas razões mencionadas, o desenvolvimento de novos protocolos, especializados para Redes de Sensores Sem Fio, para as camadas de controle de acesso ao meio, de rede e de transporte é necessário. Um ponto chave para a maioria desses tipos de rede é a auto-organização, na qual os dispositivos iniciam as atividades de formação da rede por meio do descobrimento de seus vizinhos e construção de uma tabela com informações dos vizinhos locais. Essa informação pode ser utilizada para a função de roteamento, como por exemplo, a criação de uma rota de menor distância entre o nó sensor e o nó sorvedouro. Outra alternativa para a questão do roteamento de mensagens é pelo reconhecimento, por meio do nó sorvedouro, da topologia da rede e criação de uma estrutura de roteamento.

Devido às características intrínsecas da comunicação sem fio, por exemplo, ruídos eletromagnéticos, existem várias técnicas que visam tornar a comunicação mais confiável ao nível das camadas de controle de acesso ao meio, de enlace e de transporte. Um exemplo dessas técnicas é a monitoração da qualidade do canal (link) de rádio por meio da medição da intensidade do sinal recebido e da taxa de recepção de pacotes, o que possibilita o ajuste adaptativo da potência de transmissão para manter a comunicação em níveis confiáveis. O principal objetivo é manter uma comunicação confiável com o mínimo de gasto de energia (STANKOVIC, 2008a).

2.1.4 Unidade de energia

Os nós sensores em uma Rede de Sensores Sem Fio são altamente dependentes da disponibilidade de energia e ainda possuem a expectativa de operar por períodos de tempo prolongados a partir da fonte de energia disponível no próprio nó sensor. Em função dessa necessidade, o hardware e software constituintes do nó sensor devem ser projetados com o cuidado de economizar energia em todas as etapas do projeto. A eficiência energética é um dos principais focos de pesquisa na tecnologia desse tipo de rede, porém em função das inúmeras oportunidades de utilização dessa tecnologia, surgiram aplicações nas quais as necessidades de uma maior miniaturização e de dispositivos de longa duração são requisitos mandatórios. Para equacionar essa demanda, frequentes pesquisas na busca de maior eficiência energética têm tido destaque considerável no meio acadêmico (MERRET et al., 2009).

O estudo da questão de energia pode ser realizado por meio da divisão do subsistema de energia em três componentes, a “Fontes de Energia”, o “Armazenamento de Energia” e os “Consumidores de Energia” (MERRET et al., 2009) . Esses componentes estão destacados na figura 3.

A energia é fornecida ao nó sensor por meio do componente Fontes de Energia, a qual pode ser obtida diretamente do meio físico, utilizando, por exemplo, células solares, dispositivos que geram energia elétrica quando submetidos a movimentos mecânicos e dispositivos eólicos, entre outros. Uma forma alternativa de se obter energia é por meio da utilização de uma fonte de alimentação ou através de recarga e substituição manual de baterias esgotadas. A energia obtida a partir da fonte de energia é acondicionada no componente Armazenamento de Energia, que, na prática, é constituído por uma ampla gama de tipos de baterias ou supercapacitores, dependente do tipo de aplicação. Por fim, a energia armazenada é utilizada no componente Consumidores de Energia, o qual representa os dispositivos de hardware, tais como o processador, as memórias, os sensores e o transceptor (transmissor e receptor) de ondas de rádio. Adicionalmente, outros elementos

podem ser agrupados nesse componente, como sistema de posicionamento e/ou atuadores.

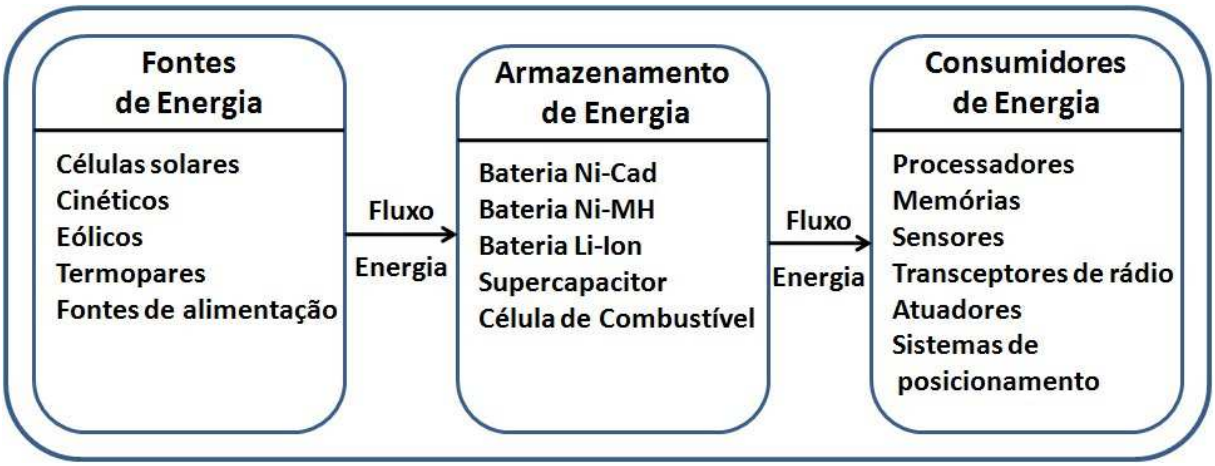


Figura 3 – Componentes típicos do subsistema de energia de um nó sensor

2.1.5 Sistema de localização

O principal objetivo de um nó sensor é o envio da informação coletada para o nó sorvedouro. Em várias ocasiões é importante para o nó sorvedouro conhecer a localização física na qual foi feita a medição da grandeza física. Por exemplo, em situações de monitoramento de incêndio. Nesse tipo de aplicação não basta apenas o conhecimento de que existe um foco de incêndio em uma região de tamanho significativo, como uma floresta, por exemplo, mas também o conhecimento da localização geográfica desse foco.

Em geral os nós sensores são lançados aleatoriamente no ambiente sob investigação e trabalham coletivamente, de forma não assistida, sem conhecimento a priori de sua localização. Sistemas de posicionamento global, como GPS, dependendo da aplicação, podem ser utilizados nessa tarefa. Por outro lado, sua utilização pode não ser viável em função do tamanho, do consumo de energia e posicionamento em áreas internas, sem visibilidade de satélites constituintes da constelação do sistema de posicionamento global. Nessas oportunidades torna-se necessária a utilização de outros meios para a estimação da posição do nó sensor.

Um método alternativo à utilização do GPS pode ser realizado por meio da utilização de um sinal de rádio, conhecido como beacon, no qual um nó em especial, geralmente o nó coordenador, conhecedor a priori de sua posição, com o conceito de âncora, emite o beacon para auxiliar aos demais nós da rede a calcularem suas posições. Esse método não é aconselhável para utilização em aplicações com um número elevado de nós e nem em condições ambientais adversas (YICK; MUKHERJEE; GHOSAL, 2008).

Além dos métodos descritos anteriormente, existe um método que utiliza a característica de proximidade dos nós sensores vizinhos. Um nó de referência, cuja posição é conhecida emite o sinal de beacon para os demais nós próximos. Uma forma de se avaliar a distância dos nós vizinhos ao nó de referência é por meio da avaliação da potência do sinal de beacon recebida, possibilitando, por trilateração, a estimação da localização dos nós sensores.

2.1.6 Atuadores

A utilização de atuadores em Redes de Sensores Sem Fio não é comum devido à própria natureza da rede que possui como funções básicas o sensoriamento, o processamento e a transmissão de dados. O papel dos atuadores é interagir com o ambiente ao seu redor no sentido de estabelecer uma nova condição desejada, e para isso necessita de muita energia. Como exemplo de atuadores pode-se citar os motores, os solenóides, os relés, os posicionadores e as eletroválvulas, entre outros.

Quando a rede é concebida para a utilização de atuadores, ela recebe uma denominação diferenciada, ou seja, Redes de Sensores e Atuadores Sem Fio – *Wireless Sensor and Actuator Network* (WSAN). Nessa condição os nós sensores permitem a abertura ou fechamento de chaves, relés ou a geração de um valor binário adequado para a ativação ou desativação dos atuadores. Um exemplo de utilização de nós atuadores, pode ser verificado no trabalho de Yeh et al. (2010) no qual é proposto um sistema autônomo de controle de luminosidade, tornando o

ambiente dinamicamente adaptável, de acordo com a preferência do usuário e proporcionando a economia de energia. Esses ambientes são conhecidos como “ambientes inteligentes”.

2.2 APLICAÇÕES DE REDES DE SENSORES SEM FIO

As Redes de Sensores Sem Fio têm sido implantadas para monitorar e disseminar, coletivamente, informação sobre uma variedade de fenômenos de interesse. Seu papel básico é efetuar o sensoriamento em rede utilizando elevado número de sensores simples, ao contrário da abordagem convencional da utilização de poucos instrumentos de monitoração sofisticados e caros. As principais vantagens da monitoração por meio da utilização de Redes de Sensores Sem Fio sobre a técnica convencional são: ampla cobertura, precisão e confiabilidade, a um custo relativamente inferior (LI; THAI; WU, 2008). É importante destacar que o conjunto de potenciais aplicações é grande e vai além das áreas militar, civil, ambiental, comercial e industrial. Como exemplo, pode-se mencionar a utilização desse tipo de redes em aplicações de: vigilância militar, monitoração e controle do processo de produção, monitoração de sinais vitais humano (batimento cardíaco, pressão arterial e temperatura entre outros), monitoração de habitat ou condições ambientais e de monitoração residencial, entre outras.

2.2.1 Características das Redes de Sensores Sem Fio

Embora as Redes de Sensores Sem Fio tenham algumas semelhanças com as redes convencionais não estruturadas, conhecidas como *Mobile Ad hoc NETwork* (MANET), existem diferenças significativas e desafios específicos a serem observados e pesquisados (KARL; WILLIG, 2003). A seguir são descritas as principais características desse tipo de redes, as quais não são comumente encontradas nas redes sem fio convencionais.

2.2.1.1 Aplicação específica

A quantidade considerável de possíveis combinações das tecnologias de sensoriamento, de processamento e de comunicação, fornece um cenário para o desenvolvimento de um grande número de aplicações diferentes. Não se tem conhecimento de uma solução única e padrão que possa abranger todas as diferentes necessidades das inúmeras aplicações. Como exemplo, pode-se citar a questão do número de nós na rede. Existem aplicações que necessitam de um número relativamente pequeno de nós sensores, sendo que essas redes são classificadas como esparsas. Outras aplicações necessitam de um número elevado de nós sensores, cuja estrutura da rede é diferente e são denominadas por redes densas. Essas duas classes de aplicações demandam protocolos de roteamento diferentes ou, até mesmo, adaptativos. O mesmo ocorre com os protocolos de controle de acesso ao meio, tendo em vista que, no segundo caso, a concorrência na obtenção de acesso ao canal de comunicação é significativamente maior que no primeiro (KARL; WILLIG, 2003).

2.2.1.2 Interação com o meio físico

A interação do nó sensor com o meio físico na qual a rede está inserida proporciona um padrão de tráfego de informações significativamente diferente das redes convencionais. Uma característica comum às Redes de Sensores Sem Fio é apresentar baixa taxa de tráfego de dados por longos períodos de tempo, podendo apresentar um aumento significativo no tráfego de dados, em função da ocorrência de algum evento monitorado, por exemplo, durante a monitoração de incêndios e enchentes.

2.2.1.3 Escalabilidade

Potencialmente, uma Rede de Sensores Sem Fio deve agregar um número de dispositivos significativamente maior do que em redes convencionais, necessitando de soluções escaláveis. A associação e a desassociação de nós sensores é comum

nessa tecnologia e a quantidade desses componentes pode variar de dezenas até milhares.

2.2.1.4 Energia

A fonte de energia é um recurso escasso em Redes de Sensores Sem Fio e possui o agravante de não poder ser substituída em muitas aplicações, sendo, em várias oportunidades, a responsável pela longevidade da rede de sensores.

2.2.1.5 Auto-organização

De forma similar às redes ad hoc, as Redes de Sensores Sem Fio possuem a característica de se autoconfigurar, formando uma rede conectada, porém com diferenças significativas em termos de tráfego, de compromisso no consumo de energia e de algoritmos de roteamento, entre outras. Por exemplo, o processo de aprendizado da posição geográfica atual do nó sensor é uma informação valiosa no processo de auto-organização.

2.2.1.6 Dependabilidade e qualidade de serviço

As Redes de Sensores Sem Fio devem implementar os conceitos de dependabilidade e de qualidade de serviço – *Quality of Service* (QoS). Ainda, não está claro como definir qualidade de serviço nesse tipo de redes. Em determinadas situações, a entrega ocasional de um único pacote pode ser o suficiente. Em outros casos, existe a necessidade da entrega confiável e de forma determinística. A taxa de entrega de pacotes é uma métrica pobre e insuficiente. O que é importante de se mensurar em uma Rede de Sensores Sem Fio é a quantidade e a qualidade da informação que pode ser extraída de determinadas áreas ou objetos observados. Novamente o quesito energia deve ser considerado, devido ao compromisso da quantidade de energia necessária para se obter essas métricas (KARL; WILLIG, 2003).

2.2.1.7 Arquitetura de redes centrada em dados

Devido à proposta de baixo custo do nó sensor e de fonte de energia escassa, é comum o lançamento desses dispositivos de forma redundante. Como consequência, a importância de um nó sensor específico é reduzida em comparação com as redes convencionais. O mais importante são os dados que esses nós podem aquisitar. Essa característica intrínseca das Redes de Sensores Sem Fio promove uma mudança no paradigma de arquiteturas de redes centradas em nós (*node centric*), ou centradas em usuários (*user centric*), para arquiteturas de redes centradas em dados (*data centric*) (KARL; WILLIG, 2003).

2.2.1.8 Simplicidade

Tendo como princípio que o nó sensor possui recursos mais limitados (capacidade de processamento, memória e energia), os softwares de controle (sistemas operacionais embarcados) e os softwares de controle de rede devem ser dimensionados apropriadamente, ou seja, possuir um grau de simplicidade significativa quando comparados com os softwares utilizados em computadores pessoais. Essa simplificação, muitas vezes, leva ao rompimento do paradigma da tradicional abordagem de interligação de computadores em rede utilizando o modelo de referência OSI (descrito no próximo capítulo), uma vez que a criação de camadas de abstração é uma atividade que consome tempo de processamento, espaço em memória e energia.

2.2.2 Classificação das Redes de Sensores Sem Fio

Cada aplicação que envolva a utilização de Redes de Sensores Sem Fio apresenta um conjunto específico de objetivos e restrições, gerando diferentes padrões de tráfegos de dados na rede. Esse é um desafio em termos de projeto de redes, impondo a necessidade de pesquisa e desenvolvimento de protocolos de comunicação específicos para cada tipo de aplicação. Esse tipo de abordagem não

é eficiente, além de necessitar de investimentos significativos. A forma de equacionar esses desafios é por meio da avaliação criteriosa entre protocolos genéricos e protocolos específicos de forma a avaliar o compromisso da escolha.

Uma maneira eficiente de abordar esses desafios é agregar as aplicações de Redes de Sensores Sem Fio em grupos e classificá-los em função de critérios comuns. Nesse sentido os autores Karl e Willig (2005) propuseram uma divisão na qual as aplicações são divididas nas seguintes classes: detecção de eventos, medições periódicas, função de aproximação e detecção de borda e, por fim, rastreamento. Essa classificação leva em conta o padrão de interação entre os nós sensores e os nós sorvedouros.

Outra proposta de classificação das aplicações de Redes de Sensores Sem Fio foi apresentada por e Li, Thai e Wu (2008), na qual os autores utilizam as seguintes características:

1. Os objetivos da aplicação;
2. As necessidades de entrega de dados; e
3. As características do tráfego de dados.

Tendo como base as duas propostas descritas é possível classificar a grande maioria das aplicações em Redes de Sensores Sem Fio em quatro grandes classes apresentadas a seguir. Essa classificação leva em conta o tipo de aplicação, o tipo de fluxo de informação e a interação entre os nós sorvedouro e sensores.

2.2.2.1 Detecção de eventos e transmissão da informação

A principal característica desse conjunto de aplicações é a ocorrência esporádica do evento de interesse. As Redes de Sensores Sem Fio para essa classe de aplicações têm a expectativa de ficarem inativas a maior parte do tempo, entrando em atividade quando o evento é detectado, momento no qual a Rede de Sensores Sem Fio tem que, prontamente, transmitir as informações sobre o evento para o nó sorvedouro.

Essa informação deve conter dados sobre a localização e também a natureza do evento.

Uma questão importante para essas redes, no nível da aplicação, é minimizar a probabilidade da geração de falsos alarmes. Esse quesito pode ser melhor trabalhado por meio da utilização de um conjunto de sensores ao invés de um único sensor. Avaliar as informações de um grupo de sensores com o objetivo de encontrar um ponto de consenso é mais confiável do que tomar a decisão suportada por um único sensor.

Do ponto de vista da rede como um todo, outra importante questão surge da necessidade do roteamento da informação, de forma dinâmica, para o nó sorvedouro. Como exemplos de aplicações dessa classe, pode-se citar a detecção de intrusos como parte de um processo de vigilância militar, detecção de comportamento anômalo ou falhas em processos industriais e detecção de incêndio em florestas, entre outros (SIMON et al., 2004).

2.2.2.2 Aquisição e transmissão da informação de forma periódica

A principal característica dessa classe de aplicações é que cada sensor deve, de forma periódica, produzir uma quantidade de informações que deverá ser encaminhada ao nó sorvedouro. O destinatário pode estar diretamente interessado em uma medida consolidada de uma grandeza de interesse monitorada pela rede, e pode necessitar da computação distribuída de alguma função, como agregação, por exemplo, dos valores aquisitados por um grupo de sensores.

Como exemplos de aplicações dessa classe, pode-se citar a monitoração das condições ambientais que afetam a produção agrícola ou pecuária, a monitoração das condições ambientais de um prédio comercial, como temperatura, umidade e luminosidade, entre outros (BURGESS et al., 2010; TOLLE et al., 2005).

Outra possibilidade interessante é a combinação de módulos atuadores ao nó sensor, com o intuito de fornecer algum nível de controle ao sistema. Dessa forma,

nessas aplicações poderia ser controlada a quantidade de fertilizante a ser aplicada ao solo, ou ainda, poderia controlar a temperatura, por meio do resfriamento ou aquecimento do ambiente de trabalho.

2.2.2.3 Consulta de informações pela aplicação

A principal característica dessa classe de aplicações é que o nó sorvedouro inicia o processo de aquisição de informações dos eventos por meio do envio de consultas para a rede. Nesse caso, ao invés do nó sensor enviar as informações ao nó sorvedouro, este requisita as informações aos nós sensores, passando a operar de forma ativa, e não mais passiva de acordo com as duas classes anteriores. Essa característica permite ao nó sorvedouro extrair informações do sistema com diferentes resoluções e granularidades, a partir de diferentes regiões espaciais, somente quando houver a real necessidade.

Como exemplos de aplicações dessa classe, pode-se citar a monitoração de um processo industrial. No caso da ocorrência de alguma situação indesejável, o nó sensor pode enviar a informação para o nó sorvedouro. Em função dessa informação o nó sorvedouro pode indagar conjuntos específicos de sensores para obter maiores informações sobre o processo e, consequentemente, sobre o evento monitorado, podendo inclusive validar a sua ocorrência. Dependendo das informações recebidas dos nós sensores indagados, o nó sorvedouro pode ativar atuadores apropriados e/ou acionar um alarme para invocar a intervenção humana. O projeto Cougar, desenvolvido por Yao e Gehrke (2002), propõe uma forma de interação entre a aplicação e a Rede de Sensores Sem Fio por meio de consultas, permitindo ao usuário obter, de forma otimizada, dados armazenados na rede. Outro trabalho nessa classe de aplicação foi desenvolvido por Cansado et al. (2010), na qual os autores propõem um processo de melhoria na segurança em Redes de Sensores Sem Fio por meio da avaliação da qualidade dos dados “colhidos” em campo.

2.2.2.4 Rastreamento

As aplicações dessa classe combinam algumas das características descritas nas classes anteriores. Quando um alvo é detectado, o nó sorvedouro deve ser imediatamente notificado. Nesse momento o nó sorvedouro deve iniciar uma série de questionamentos aos nós sensores envolvidos, visando a recepção de informações sobre o momento e o local do objeto rastreado. De posse dessas informações o nó sorvedouro pode calcular a trajetória estimada do objeto e iniciar a consulta ao grupo de sensores próximos ao local de trajeto do objeto.

Como exemplos de aplicações dessa classe, pode-se citar as militares ou vigilância de fronteiras, nas quais existe o interesse no rastreamento de um intruso ou, ainda, o interesse no movimento de um objeto suspeito. Em aplicações ambientais pode-se citar a necessidade em se monitorar os movimentos de insetos, animais e objetos (JUANG, 2002). Em sistemas de transporte inteligente o rastreamento dos veículos em grandes áreas permite a extração de informações estatísticas, as quais podem ser utilizadas na melhora da infraestrutura rodoviária e também evitar acidentes por meio de alertas aos motoristas (XING, 2005).

2.3 DESAFIOS EM REDES DE SENSORES SEM FIO

De acordo com as informações relatadas nas seções anteriores, percebe-se que a abordagem isolada das Redes de Sensores Sem Fio não é a mais adequada. Essas redes são fortemente acopladas com a aplicação e, embora existam inúmeras ferramentas de análise desenvolvidas para redes convencionais, elas não são apropriadas para a utilização na solução dos problemas em Redes de Sensores Sem Fio. O desenvolvimento de ferramentas adequadas aos diversos tipos de aplicações descritas ou adaptações em ferramentas tradicionais existentes é um desafio considerável a ser transposto. Embora alguns itens relacionados a seguir sejam comuns aos abordados na seção anterior, o foco presente é relacionar os desafios e requisitos comuns a serem observados no desenvolvimento dessas redes.

- Tolerância a Falhas: Considerando que os nós sensores podem apresentar falhas de diversos tipos (energia, comunicação, etc.), é conveniente dotar as Redes de Sensores Sem Fio com algum tipo de redundância. Esse desafio pode ser contornado por meio do lançamento de um número de elementos maior que o estritamente necessário caso todos eles operassem corretamente. O excesso de nós sensores é capaz de proporcionar um nível de redundância que permita um grau adequado de tolerância a falhas.

- Escalabilidade: Uma Rede de Sensores Sem Fio típica pode envolver uma quantidade considerável de nós sensores, e pode variar de algumas dezenas até alguns milhares de elementos. Logo, é importante que as arquiteturas e os protocolos de redes desenvolvidos tenham a habilidade de lidar com essa gama de elementos componentes desse tipo de rede, sempre permitindo variação do número de nós sensores.

- Tipos de Serviços: O serviço básico oferecido pelas redes de comunicação digitais convencionais consiste, basicamente, em que a rede transmite um bit – *BInary Digit* (BIT) de um determinado lugar (transmissor) para outro (receptor). Em Redes de Sensores Sem Fio a transmissão do referido bit também é necessária, porém não é a única tarefa. Muitas vezes, esses tipos de redes devem fornecer informações adicionais, como exemplo, as informações geográficas e temporais referentes aos dados transmitidos. Novos paradigmas para o uso dessas redes necessitam ser desenvolvidos, bem como novas interfaces e novas maneiras de oferecimento de serviços.

- Densidade Variável: A densidade das Redes de Sensores Sem Fio, definida como número de nós sensores em uma determinada área, pode variar significativamente. Essa variação pode ocorrer em função da necessidade da aplicação, pois aplicações diferentes necessitam de densidades diferentes. Mesmo em uma determinada rede, a densidade pode variar em função do tempo e do espaço, devido às falhas de nós sensores, ou ainda a mudança da posição do dispositivo. É importante notar que a densidade da rede não é homogênea, devido, por exemplo, ao lançamento aleatório. Portanto, esse tipo de rede possui o desafio de se adaptar a essas variações de densidade.

- Vida Útil: Em diversos cenários a Rede de Sensores Sem Fio deve operar pelo menos o tempo necessário para executar a atividade proposta. Devido à pequena disponibilidade de energia, a vida útil da rede passa a ser um ponto importante nos projetos desse tipo de rede. Nesse sentido, modos de operações que tenham eficiência energética maior são recomendados. Uma opção atrativa para aumentar a vida útil da rede é por meio da utilização de fontes de energia renováveis, o que permitiria condições de operação mais favoráveis.

  Existe um forte compromisso entre qualidade de serviço e vida útil da rede. A utilização de uma quantidade maior de energia pode levar a um aumento na qualidade de serviço, em detrimento à vida útil da rede. Técnicas que harmonizem esses compromissos são necessárias.

  A definição de vida útil por sua vez depende da aplicação. A primeira opção é a mais simples, pois considera como vida útil o tempo decorrido entre a ativação da rede e a ocorrência da primeira falha em qualquer dos nós sensores. A segunda opção, leva em conta o tempo decorrido entre a ativação e a ocorrência da divisão da rede em dois ou mais ramos devido à ocorrência de falhas. Uma terceira opção poderia ser o tempo decorrido até que 50% dos nós sensores tenham falhado, e uma quarta opção que seria o tempo decorrido até que uma determinada região de interesse não possa mais ser monitorada por um único sensor sequer. Essas definições de vida útil não esgotam todas as opções disponíveis e tentam apenas elucidar o fato que o tempo de operação da Rede de Sensores Sem Fio pode ser quantizado de diferentes formas em função da necessidade da aplicação.

- Qualidade de Serviço: Segundo Alves et al., (2009) a qualidade de serviço (QoS) é tradicionalmente associada a taxa de transferência de bits e dados, capacidade de tráfego da rede, atrasos no envio e recebimento de mensagens e a taxa de erro no envio e recebimento de bits e pacotes. Embora essas métricas sejam significativas em redes convencionais, elas possuem importância relativa para Redes de Sensores Sem Fio. De acordo com os

autores, outras propriedades não funcionais, como escalabilidade, segurança ou sustentabilidade em termos de energia, devem ser levadas em conta no projeto dessas redes. Existem aplicações em que o envio de um único pacote de dados é o suficiente. Em outros casos, a transmissão de dados com alta confiabilidade se faz presente. Nas aplicações de tempo real o atraso desempenha um papel limitante, pois os atuadores são controlados por uma Rede de Sensores Sem Fio. A taxa de entrega de pacotes é uma métrica insuficiente, sendo que o relevante é a quantia e a qualidade da informação que pode ser extraída pelo nó sorvedouro sobre um determinado objeto ou uma região.

Por exemplo, a confiabilidade da detecção de eventos ou o fornecimento de um mapa espacial e temporal da ocorrência do evento podem ser utilizados como métricas de qualidade, com uma maior efetividade do que a taxa de erro de bit da rede – *Bit Error Rate* (BER).

- Requisitos Temporais: Com a proliferação da tecnologia de Redes de Sensores Sem Fio, será inevitável sua utilização em aplicações com restrições temporais, por exemplo, nas áreas de controle de processo e automação industrial, nas quais a monitoração do fenômeno de interesse e o controle de atuadores são tarefas com fortes dependências temporais. Nessas condições é desejável que cada nó sensor realize uma quantidade razoável de computação em tempo real. A taxa de amostragem de uma grandeza variável é significantemente maior em aplicações com restrições temporais, do que em redes convencionais para o monitoramento de ambiente, chegando, por exemplo, a algumas centenas de Hertz (CHITNIS et al., 2007). Portanto, é necessário o desenvolvimento de mecanismos que abordem as questões temporais, como a utilização ou não de sistemas operacionais dedicados com funcionalidades de garantias temporais, tratamento dos protocolos de comunicação de forma eficiente em termos de atrasos e formas de acesso ao meio de comunicação que garantam o envio e recebimento das informações em tempos previsíveis.

- Ferramentas de Simulação: A falta de ferramentas de simulação adequadas para avaliação das características das Redes de Sensores Sem Fio de forma

integrada é mais um desafio para essa emergente tecnologia. A simulação desempenha um papel importante, como no caso da comparação da influência de diferentes algoritmos, por exemplo, de escalonamento de tarefas para o processador, e também dos diversos protocolos, no desempenho global do sistema. Em automação industrial a simulação é importante na avaliação do sistema antes de sua implantação física, no sentido de possibilitar a verificação do desempenho, do correto dimensionamento e, também, para identificar, prematuramente, possíveis problemas que, porventura, não tenham sidos levados em consideração durante o projeto.

O que existe disponível atualmente são ferramentas dispersas que abordam determinadas características da rede de maneira individual, sendo necessário o exercício de escolha de ferramentas que mais se aproximem da real necessidade, ou da utilização de diversos ambientes para o desenvolvimento da plataforma de simulação desejada. Vale ressaltar que durante o presente trabalho, não foi identificada uma ferramenta integrada que atenda às necessidades de RSSF, e que possibilite o estudo e o desenvolvimento de aplicações variadas de forma integrada, segura, previsível e confiável.

2.4 MANUTENÇÃO DA QUALIDADE DA CONEXÃO EM REDES DE SENSORES SEM FIO

Em Redes de Sensores Sem Fio alguns fatores influenciam a manutenção da qualidade da conexão, e consequentemente, a confiabilidade do sistema como um todo. A seguir são detalhados alguns fatores importantes a serem mensurados em aplicações envolvendo Redes de Sensores Sem Fio.

A monitoração do nível de tensão da bateria de cada nó participante da rede é importante na manutenção da comunicação. Um nó sensor, cuja bateria esteja com baixa carga pode comprometer a operação, sendo prudente a monitoração no nível de tensão da bateria com o intuito de diminuir seus efeitos negativos, seja na substituição da bateria, quando a aplicação assim permita, seja no tratamento

diferenciado dos dados coletados em função do baixo nível de energia. Outra métrica relevante envolvendo o consumo energético é a avaliação da expectativa da vida útil do nó sensor em função do consumo de energia. Este índice pode ser calculado estimando-se o tempo de duração da carga da bateria do nó sensor devido ao gasto de energia com a operação da rede.

A temperatura de operação do nó sensor, (distinta da temperatura monitorada) também é um fator que deve ser levado em consideração no projeto. Nós sensores operando em temperaturas extremas podem apresentar comportamento não confiável. O conhecimento da temperatura de trabalho do nó sensor é uma informação valiosa para o correto funcionamento da rede. Temperaturas monitoradas acima da temperatura de operação podem danificar o nó.

A taxa de recepção de pacote – *Packet Reception Rate* (PRR) é um índice útil na determinação da taxa de pacotes efetivamente recebidos com sucesso. Ela pode ser calculada pelo controlador remoto, como sendo a relação entre o número de pacotes recebidos com sucesso e o número total de pacotes efetivamente enviados. O número de pacotes perdidos pode ser determinado através do registro de um número sequencial em cada pacote enviado.

O tempo de resposta é um indicativo do comportamento temporal do sistema, sendo recomendado, principalmente, em aplicações que apresentam restrições temporais. Ele é utilizado como métrica para o levantamento da influência de diversas condições da rede. Em geral, o tempo de resposta (TR) definido como sendo o intervalo de tempo entre a ocorrência efetiva do evento monitorado (TE) e o momento de detecção do evento (TD) pelo controlador remoto.

Existem algumas formas de se estimar o tempo de resposta, como por exemplo, por meio do cálculo do tempo de ida-e-volta. Nesse método, o sorvedouro envia uma mensagem de teste para a rede a qual é prontamente reenviada pelos nós sensores ao emissor. O tempo utilizado na operação pode ser utilizado como um estimador do tempo de resposta. Outra forma de se avaliar o tempo de resposta é através da sincronização da rede e cada mensagem que trafegasse na rede conteria o registro do instante em que a mesma foi gerada.

3 PADRÕES PARA REDES DE SENSORES SEM FIO

Este capítulo apresenta os conceitos sobre as padronizações utilizadas em Redes de Sensores Sem Fio, tendo como ponto de partida o modelo de referência OSI. As camadas inferiores (física e controle de acesso ao meio) do modelo são padronizadas pelo IEEE e as camadas superiores (aplicação e rede) são padronizadas pela aliança ZigBee™.

3.1 MODELO DE REFERÊNCIA OSI

Uma forma interessante de abordar uma tecnologia emergente, como no caso das Redes de Sensores Sem Fio, é por meio da utilização de um modelo de referência tradicional, e a partir deste avaliar as diferenças e semelhanças com a tecnologia convencional. Em termos de padronização da comunicação entre computadores, a Organização Internacional para Padronização – *International Organization for Standardization* (ISO) foi pioneira, por meio da criação do modelo de referência para interconexão de sistemas abertos, denominado de modelo OSI. O modelo de referência OSI divide as redes de computadores em sete camadas de abstração, conforme pode ser observado na figura 4.

Figura 4 – Modelo de referência OSI

A primeira camada, denominada de “física” é a mais inferior do conjunto de protocolos de comunicação, e é responsável pela transmissão dos bits através do meio físico, pela definição das características elétricas (níveis de tensões e tipos de modulações utilizadas, entre outros) e mecânicas (interface e pinagem), pelo controle da velocidade e da quantidade de bits transmitidos. Em síntese, essa camada deve garantir que quando o transmissor envia o bit “1” o receptor receba e reconheça o bit como sendo “1” e não “0”.

A segunda camada, denominada de “acesso ao meio”, também é conhecida como conexão de dados e ainda de controle de acesso ao meio. A principal funcionalidade dessa camada é transformar um canal de transmissão bruto em um meio livre de erros para a camada superior, ou seja, para a camada de rede. Essa funcionalidade é obtida por meio da divisão dos dados de entrada em quadro de dados (conjunto de bytes) e transmissão sequencial ao receptor. Em redes de difusão, as quais possuem um meio de comunicação compartilhado, tem-se o agravante que é o controle do acesso a esse canal, que deve ser equacionado por essa camada (TANENBAUM, 2003).

A terceira camada, denominada de “rede”, tem como função principal determinar a maneira com que os quadros de dados (datagramas) são roteados da origem até o destino. Essas rotas podem ser baseadas em tabelas estáticas, raramente alteradas ou dinâmicas, que são atualizadas em função de possíveis alterações na topologia da rede. Essa camada também é responsável pelo controle de congestionamento, ou seja, pelo controle de envio de um grande número de pacotes ao mesmo tempo, pelo mesmo caminho, o que fatalmente ocasionaria um ponto de gargalo. Outra questão abordada nessa camada refere-se à qualidade do serviço fornecido pela rede, devido a retardos, tempos em trânsito e instabilidade, entre outros.

A quarta camada, denominada de “transporte”, cuja principal atividade é aceitar os dados da camada de sessão, definida a seguir, dividi-los em pacotes, caso seja necessário, e enviar esses pacotes à camada de rede e garantir que todos os pacotes enviados chegarão corretamente no receptor. Outro papel importante é o isolamento proporcionado às camadas superiores em termos de alteração de tecnologias de hardware, ou seja, as camadas de sessão, de apresentação e de

aplicação, ficam imunes a qualquer alteração que porventura ocorra nas camadas inferiores. Essa camada é uma camada ponto-a-ponto autêntica, que liga a origem ao destino, ou seja, permite a comunicação entre os programas que estão sendo executados em máquinas diferentes. Enquanto os níveis inferiores tratam da comunicação entre nós adjacentes, a camada de transporte é responsável por criar um ambiente de comunicação confiável entre a origem e o destino, independente do número de nós entre o transmissor e o receptor da mensagem (TANENBAUM, 2003).

A quinta camada, denominada de “sessão”, é responsável por estabelecer mecanismos pelos quais duas aplicações sendo executadas em computadores diferentes estabeleçam uma sessão de comunicação. Para atingir esse objetivo, essa camada disponibiliza diversos serviços, entre eles, o controle de diálogo (quem transmite em determinado momento), o gerenciamento de *token* (evitar que as duas partes executem operações críticas ao mesmo tempo) e a sincronização (mecanismo em que transmissões de longa duração sejam recuperadas no caso de ocorrência de alguma falha).

A sexta camada, denominada de “apresentação” trata da sintaxe e a semântica das informações transmitidas, o que permite a comunicação entre computadores com diferentes formas de representações de dados.

A sétima camada, denominada de “aplicação”, é a camada mais superior do modelo de referência OSI. Ela é responsável pela disponibilidade de uma série de protocolos comumente necessários para que os usuários estabeleçam a comunicação através da rede. Como exemplo, pode-se mencionar o protocolo – *Hiper Text Transfer Protocol* (HTTP) que constitui a base à ampla rede mundial – *World Wide Web* (WWW) e permite aos usuários a obtenção de informações disponibilizadas na rede.

Como pode ser observado, cada camada possui características particulares e trata de um conjunto de questões específicas. Dessa forma, uma determinada camada “esconde” para as camadas superiores as “dificuldades” das camadas inferiores. A prática de se adotar modelos e abstrações é uma prática comum no tratamento de sistemas complexos. Como exemplo, pode-se citar abordagem semelhante utilizada

no estudo da forma de organização estruturada de computadores (TANENBAUM, 2001).

Em virtude das particularidades das Redes de Sensores Sem Fio pode-se utilizar o modelo de referência OSI como um guia de estudo por meio de sua comparação e avaliação dos pontos comuns, e também dos pontos divergentes. Essa análise proporciona uma visão concisa dessa nova tecnologia.

Para Redes de Sensores Sem Fio o relacionamento entre as diversas camadas é mais suave, sendo que uma determinada camada pode acessar camadas não necessariamente adjacentes. Essa característica simplifica e suaviza, em termos computacionais, o processo de comunicação. Porém, essa simplificação tem seu custo, como por exemplo, falta de padronização e protocolos pobres, sem as funcionalidades desejáveis, essas funcionalidades são onerosas em termos dos recursos limitados dos nós sensores.

A grande maioria das aplicações que utilizam a tecnologia de Redes de Sensores Sem Fio tem adotado as padronizações desenvolvidas para redes pessoais sem fio de baixo tráfego. Essas padronizações contornam, de forma racional e eficiente, as dificuldades apresentadas, devido à utilização dos meios sem fio e também dos escassos recursos, possibilitando o desenvolvimento de trabalhos complexos de uma maneira mais amigável. A figura 5 apresenta a divisão da padronização das camadas de protocolo adotadas por boa parte das Redes de Sensores Sem Fio. O padrão IEEE Std 802.15.4™-2006 LR-WPAN especifica as camadas física e de controle de acesso ao meio (MAC), enquanto o padrão ZigBee™-2007 adiciona a camada de rede, fornece serviços de segurança de dados e também proporciona uma interface apropriada para a camada de aplicação. As seções a seguir descrevem com maiores detalhes os dois padrões.

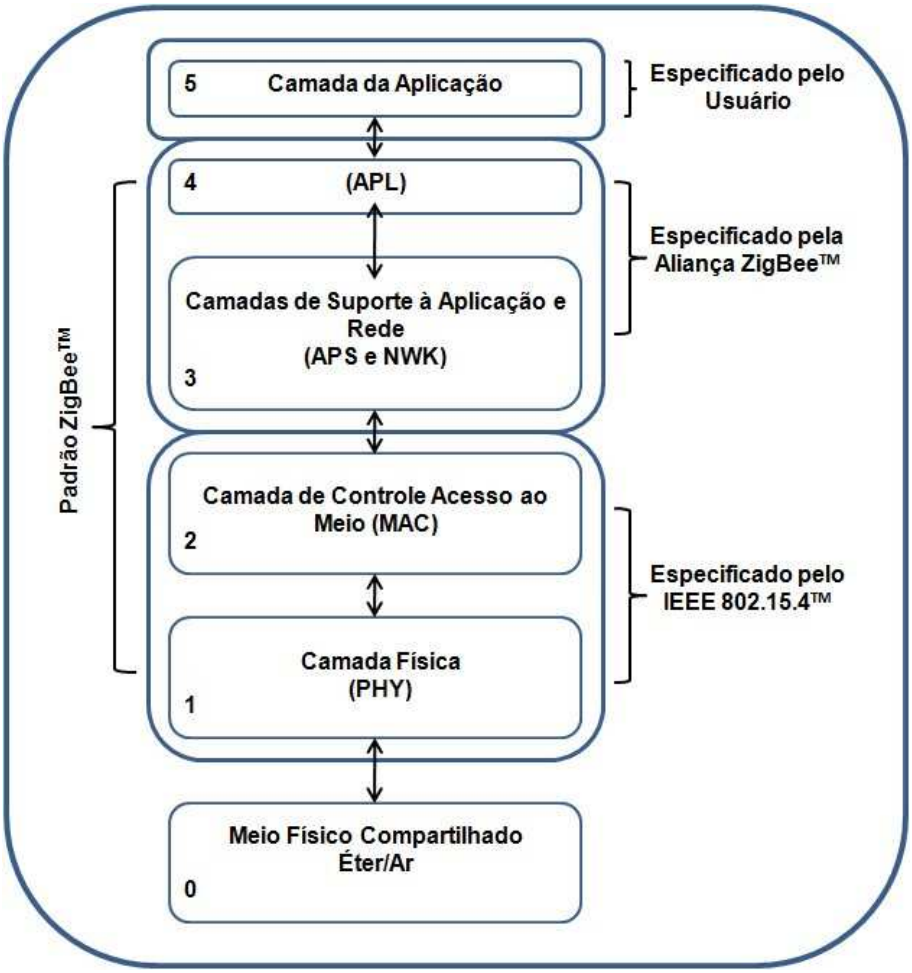


Figura 5 – Camadas de protocolos de Redes de Sensores Sem Fio

3.2 PADRÃO IEEE STD 802.15.4™-2006 LR-WPAN

O padrão IEEE Std 802.15.4™-2006 LR-WPAN define as características das camadas física (PHY) e controle de acesso ao meio (MAC) para redes locais pessoais sem fio de baixo tráfego LR-WPAN (IEEE, 2006). Este padrão aborda a normalização das camadas física e controle de acesso ao meio e em várias oportunidades, é denominado como protocolo 802.15.4. As principais vantagens de uma rede pessoal sem fio de baixo tráfego são: facilidade de instalação, transferência de dados de forma confiável, operação em curta distância, baixo custo e um baixo consumo energético. Possui também um modelo de camadas de protocolos simples e flexível.

3.2.1 Camada física

A camada física (PHY) do protocolo IEEE 802.15.4™-2006 LR-WPAN, identificada na figura 5 pela camada 1, suporta três faixas de frequência, a faixa de 2.450 MHz com um total de 16 canais, faixa esta não licenciada para uso no Brasil, e também conhecida como faixa *Industrial, Scientific and Medical* (ISM). Adicionalmente, o protocolo, fornece suporte para as faixas de 915 MHz, composta por 30 canais e também de 868 MHz, composta por 3 canais. Para todos os canais é disponibilizada a técnica de comunicação em espalhamento espectral por sequência direta – *Direct Sequence Spread Spectrum* (DSSS), e opcionalmente, para os canais das faixas de 868 e 915 MHz é disponibilizada a técnica de espalhamento espectral por sequência paralela – *Parallel Sequence Spread Spectrum* (PSSS).

Existem três técnicas de modulação disponíveis, a primeira, utilizada nas faixas de 868 e 915 MHz é conhecida como chaveamento de fase binário – *Binary Phase-Shift Keying* (BPSK), a segunda, utilizada opcionalmente nessas faixas é conhecida por chaveamento por desvio de amplitude – *Amplitude Shift Keying* (ASK) e a terceira técnica, utilizada na faixa de 2.450 MHz é denominada de chaveamento por deslocamento de fase por compensação de quadratura – *Offset Quadrature Phase-Shift Keying* (O-QPSK), esta técnica modula 4 bits (denominado de símbolo) por vez. O quadro 3 contém um resumo das características de cada faixa de frequência.

PHY frequência (MHz)	**Faixa Frequência (MHz)**	**Parâmetros / Dados**				**Duração Símbolo (µs)**	**Região Adotado**
		Modu-lação	**Taxa bit Kb/s**	**Taxa Símbolos Ks/s**	**Símbolos**		
868 - 915	868-868,6	BPSK	20	20	Binário	50	USA
	902-928	BPSK	40	40	Binário	25	Europa
868 - 915 opcional	868-868,6	ASK	250	12,5	20-bit PSSS	80	USA
	902-928	ASK	250	50	5-bit PSSS	20	Europa
868 - 915 opcional	868-868,6	O-QPSK	100	25	16 Ortogonal	40	USA
	902-928	O-QPSK	250	62,5	16 Ortogonal	16	Europa
2450	2400-2483,5	O-QPSK	250	62,5	16 Ortogonal	16	Outras regiões / Brasil

Quadro 3 – Principais características da camada física do padrão LR-WPAN

A camada física é responsável pelas seguintes atividades:

- Controlar a ativação e desativação do transceptor de rádio;
- Medir o nível de energia detectada – *Energy Detection* (ED) no canal selecionado;
- Estimar o indicador da qualidade da conexão – *Link Quality Indicator* (LQI) para os pacotes recebidos;
- Avaliar se o canal utilizado está livre – *Clear Channel Assessment* (CCA) para suporte ao método de transmissão conhecido como *Carrier Sense Multiple Access With Collision Avoidance* (CSMA-CA);
- Selecionar a frequência do canal;
- Transmitir e receber dados.

Em termos de confiabilidade, a camada física fornece algumas métricas importantes para a sua avaliação. Entre as métricas disponibilizadas destacam-se a taxa de erro de bit (BER) e a taxa de erro de pacote (PER). O padrão IEEE 802.15.4™-2006 LR-WPAN define que a camada física disponibilize, para cada pacote recebido, duas métricas diferentes, sendo uma a Energia Detectada (ED) e a outra o Indicador da Qualidade da Conexão (LQI).

A taxa de erro de bit (BER) representa a relação entre o número total de bits incorretos recebidos e o número total de bits transferidos durante o intervalo de tempo de observação em uma transmissão digital. Os bits incorretos podem ter sido alterados em função de erros provocados por ruído, interferência, distorção, problemas de sincronização, atenuação e o enfraquecimento do sinal devido ao multicaminho, entre outros, presentes no canal de comunicação utilizado. A taxa de erro de bit (BER) é adimensional e geralmente expressa em porcentagem. Este índice pode ser considerado como uma aproximação da estimativa da probabilidade da ocorrência de erros de bits em uma conexão. Na equação 5, o termo $Nerr$ representa o número de bits com erro e o termo $Nbits$ representa o número total de bits transmitidos durante o período de observação.

$$BER = Nerr/Nbits \qquad \ldots\ldots (5)$$

A taxa de erro de pacote (PER) é a relação entre o número de pacotes incorretos recebidos e o número total de pacotes transferidos. Um pacote é considerado incorreto se pelo menos um bit, constituinte do pacote, estiver incorreto. Segundo Lee, Jeong e Lee (2010), este índice pode ser calculado para um pacote contendo "N" bits de acordo com a equação 6. Este é o valor esperado da probabilidade de erros de pacotes.

$$PER = 1 - (1 - BER)^N \qquad \ldots\ldots (6)$$

O indicador de energia detectada (ED) é uma importante métrica da confiabilidade da conexão estabelecida. É o nível de energia presente no canal de comunicação selecionado. Um nível forte do sinal proporciona uma comunicação estável e, praticamente, sem perdas. O valor da energia detectada é utilizado pela camada de rede como parte do algoritmo de seleção do canal de comunicação. A especificação dos valores máximo, mínimo e o medido, para cada nó da rede, possibilita a supervisão das condições de comunicação, ou seja, do canal de comunicação escolhido. Vale a pena salientar que esse indicador não entra no mérito se a energia detectada é referente a um sinal no padrão IEEE 802.15.4™-2006 LR-WPAN.

O indicador da qualidade da conexão (LQI) permite avaliar a qualidade do pacote de dados recebido. O padrão recomenda que essa medida seja calculada utilizando a energia detectada (ED), a estimação da relação entre o sinal e o ruído ou a combinação desses dois métodos. Quando um pacote de dados corrompido é recebido, o LQI permite identificar se o problema foi devido à recepção de um sinal fraco, de baixa potência, ou um sinal forte, porém, adicionado com interferências. Os valores mínimos e máximos do LQI devem ser associados, respectivamente, com a menor e a máxima qualidade do sinal. Qualquer valor entre o máximo e o mínimo deve ser uniformemente distribuído entre eles.

O formato da unidade de dados do protocolo da camada física – PHY *Protocol Data Unit* (PPDU) é apresentado na figura 6. O pacote PPDU é constituído por três campos: O cabeçalho de sincronização – *Synchronization Header* (SHR) o qual permite ao dispositivo receptor sincronizar o fluxo binário com o transmissor. O cabeçalho físico – *PHY header* (PHR), que contém informações sobre a quantidade

de dados transmitidos e/ou recebidos, e a carga útil (PHY payload) variável que contém o quadro completo da camada de controle de acesso ao meio (MAC), ou seja, o *MAC Protocol Data Unit* (MPDU), descrito na próxima seção. É importante destacar o significado dos subcampos: preâmbulo, utilizado para sincronizar o transceptor de rádio, *Start-of-Frame Delimiter* (SFD) utilizado para indicar o fim do campo SHR e *PHY Service Data Unit* (PSDU) que contém a carga útil da camada física.

Figura 6 – Formato do pacote PPDU

3.2.2 Camada controle de acesso ao meio

A camada controle de acesso ao meio (MAC), identificada na figura 5 pela camada 2, é responsável pela definição de dois tipos de dispositivos. Os dispositivos com funções reduzidas, *Reduced-Function Device* (RFD), são caracterizados por possuírem poucos recursos e somente operam como elementos terminais da rede, os quais, em Redes de Sensores Sem Fio, são habitualmente denominados de nós sensores. Já os dispositivos com funções completas, *Full-Function Device* (FFD), são mais complexos e implementam todo o conjunto das funções da camada de acesso ao meio (MAC), o que os habilita a agirem como nós coordenadores ou dispositivos terminais da rede. Em Redes de Sensores Sem Fio podem desempenhar os papéis de nós sensores, de nós sorvedouros, de nós roteadores, ou ainda, de nós clusters. Quando configurado como coordenador de rede o dispositivo FFD possui a capacidade de enviar um sinal especial denominado de *beacon*, fornecendo serviços de sincronismo, de comunicação e de agregação.

Existem, basicamente, dois tipos de topologias de redes padronizadas para a camada MAC. Na topologia em estrela é utilizado o modelo mestre-escravo, no qual um dispositivo FFD é designado como coordenador da rede local pessoal – *Personal Area Network* (PAN) e os escravos são dispositivos RFDs ou FFDs que podem somente se comunicar com seu coordenador, estabelecendo uma rede denominada de um único-salto (*single-hop*). A principal vantagem desse tipo de topologia é a sua simplicidade, pois a comunicação é centralizada entre o coordenador da rede e os dispositivos associados. Na topologia ponto-a-ponto, um dispositivo FFD pode se comunicar com outros dispositivos FFDs fora de sua cobertura de rádio por meio de outros nós FFDs intermediários, formando uma rede multi-saltos (*multi-hop*). A operação completa é gerenciada pelo coordenador da rede pessoal (PAN). A figura 7 exemplifica as topologias de redes suportadas pela camada controle de acesso ao meio (MAC) do padrão IEEE 802.15.4^{TM}-2006 LR-WPAN.

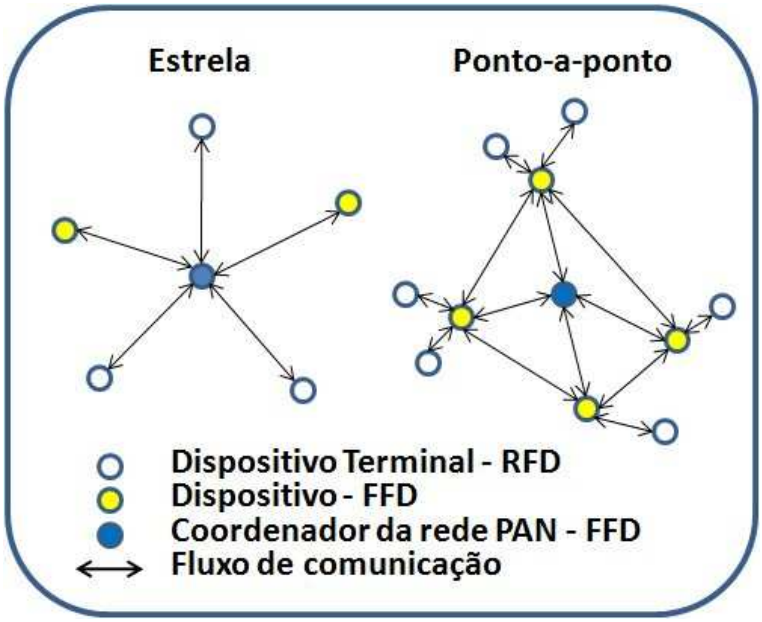


Figura 7 – Topologias estrela e ponto-a-ponto

A camada controle de acesso ao meio (MAC) especifica uma estrutura denominada de quadro que possui três campos principais, o primeiro denominado de cabeçalho *MAC header* (MHR), o segundo como sendo carga útil (*MAC payload*) e, por último, o *MAC footer* (MFR). O quadro completo da camada MAC é identificado por MPDU e composto pelos três campos citados. Esse quadro completo é a carga útil para a

camada física, onde recebe a denominação de PSDU, conforme pode ser verificado na figura 6.

O padrão IEEE 802.15.4^{TM}-2006 LR-WPAN especifica quatro tipos de quadros dependendo de sua função.

3.2.2.1 Quadro de beacon

É um quadro emitido periodicamente pelo coordenador com o objetivo de manter a rede sincronizada, possibilitando mecanismos de chaveamento de períodos de operação ativos e inativos, melhorando a eficiência em termos de consumo de energia. Esse quadro, além de identificar a rede também especifica a estrutura do superquadro. O quadro de beacon é gerado automaticamente na camada de acesso ao meio. A estrutura do quadro de beacon é apresentada na figura 8.

- O campo *MHR* é composto de quatro sub-campos, o controle do quadro, o número de sequência do beacon, o endereçamento e o cabeçalho auxiliar de segurança de dados opcional. O campo endereço pode ter dois comprimentos em função do tipo de endereço utilizado para identificação dos dispositivos da rede. O padrão IEEE 802.15.4^{TM}-2006 especifica dois tipos de endereços, o primeiro, denominado de endereço curto, utiliza apenas 2 bytes, o segundo, denominado de endereço longo, utiliza 8 bytes e adicionalmente ao endereço é especificado também a identificação da rede com 2 bytes. O cabeçalho auxiliar de segurança é variável em função da utilização ou não dos níveis de segurança de dados disponíveis, conforme relacionados no quadro 4 apresentado na próxima sessão.

- O campo *Mac payload* é composto por quatro sub-campos, sendo o primeiro responsável pela especificação do formato do superquadro, descrito na subseção 3.2.2.4, o segundo, de tamanho variável, contém os compartimentos de tempo garantidos – *Guaranteed Time Slot* (GTS), o terceiro contém os endereços pendentes e o quarto contém a carga útil do beacon.

- O campo *MFR* é formado por apenas um campo denominado de verificação da sequência do quadro – *Frame Check Sequence* (FCS) composto por 2 bytes. O campo FCS é calculado por meio de um algoritmo de verificação de redundância cíclica – *Cyclic Redundancy Check* (CRC) conforme padrão especificado pelo *International Telecommunication Union – Telecommunication Standardization Sector* (ITU-T). O algoritmo é aplicado nos campos MHR e no MAC payload.

Figura 8 – Estrutura do quadro de beacon – IEEE 802.15.4™-2006 LR-WPAN

3.2.2.2 Quadro de dados

Este quadro é gerado pelas camadas superiores, por exemplo, a camada de aplicação, com o objetivo de transmitir dados pela rede. A figura 9 ilustra a estrutura do quadro de dados. Os campos MHR e MFR possuem as mesmas características do quadro descrito anteriormente, porém o campo MAC payload é reservado integralmente para dados fornecidos pelas camadas superiores.

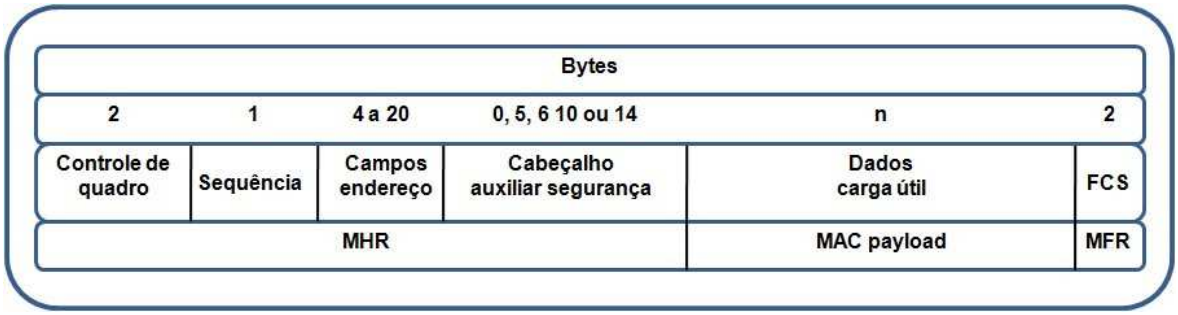


Figura 9 – Estrutura do quadro de dados – IEEE 802.15.4™-2006 LR-WPAN

3.2.2.3 Quadro de confirmação

Este quadro é gerado pela camada de acesso ao meio no sentido de possibilitar a comunicação entre os dispositivos da rede com confirmação de entrega. A figura 10 ilustra a estrutura do quadro de confirmação. Esse quadro é mais simples, composto por apenas 2 campos, o MHR simplificado e o MFR.

Figura 10 – Estrutura do quadro de confirmação – IEEE 802.15.4^{TM}-2006 LR-WPAN

3.2.2.4 Quadro de comando

Este quadro é gerado pela camada de acesso ao meio e contém comandos a serem enviados para a rede. A figura 11 ilustra o quadro de comandos. Esse quadro é semelhante aos dois primeiros quadros descrito anteriormente, ou seja, quadro de beacon e quadro de dados, sendo a diferença no campo MAC payload, no qual para o quadro de comando é utilizado para especificar o tipo de comando e os parâmetros desses comandos.

Figura 11 – Estrutura do quadro de comandos – IEEE 802.15.4^{TM}-2006 LR-WPAN

O padrão IEEE 802.15.4™-2006 LR-WPAN fornece mecanismos de suporte para aplicações de tempo real, por meio da disponibilidade do uso de compartimentos de tempo garantidos (GTS), no qual uma largura de banda pode ser reservada com exclusividade para transmissão de dados por dispositivos que possuam restrições temporais. Infelizmente o número máximo de GTS é reduzido e suporta, no máximo, apenas sete dispositivos simultaneamente. Dependendo da aplicação da rede esse número de GTS pode não ser suficiente para atender as necessidades impostas. Portanto é importante a utilização desse recurso escasso de forma racional e otimizada (SEMPREBOM et al., 2010).

A camada de controle de acesso ao meio (MAC) suporta dois tipos de operações selecionadas pelo coordenador da PAN: a primeira habilita o sinal de beacon e a segunda desabilita o sinal de beacon, sendo que o acesso ao meio compartilhado é feito por contenção, baseado na tecnologia CSMA-CA, com compartimento de tempo definido (*slotted*) ou sem definição do compartimento de tempo (*unslotted*).

No modo em que o sinal de beacon está habilitado, quadros de beacon são periodicamente enviados pelo coordenador da PAN em intervalos de tempo fixos denominados de *Beacon Interval* (BI), com os objetivos de sincronizar os dispositivos componentes da rede, de identificar a rede e de especificar a estrutura do superquadro adotada. A figura 12 ilustra a estrutura do superquadro identificando seus principais itens componentes. Como pode ser notado o superquadro é composto de 2 períodos, denominados de ativo e inativo. O período ativo, que corresponde à duração do superquadro, denominado de *Superframe Duration* (SD), é dividido igualitariamente em 16 compartimentos de tempo, os quais proporcionam a comunicação de dados. Cada período ativo, pode ainda ser subdividido em dois subperíodos, sendo que o primeiro habilita o acesso ao meio por contenção e é denominado por *Contention Access Period* (CAP), o qual utiliza a técnica CSMA-CA e o segundo, não possui contenção, ou seja, o acesso ao meio é garantido. Esse sub-período é conhecido como *Contention Free Period* (CFP) e utiliza a técnica denominada de compartimento de tempo garantido (GTS) (JURCIK et al., 2007).

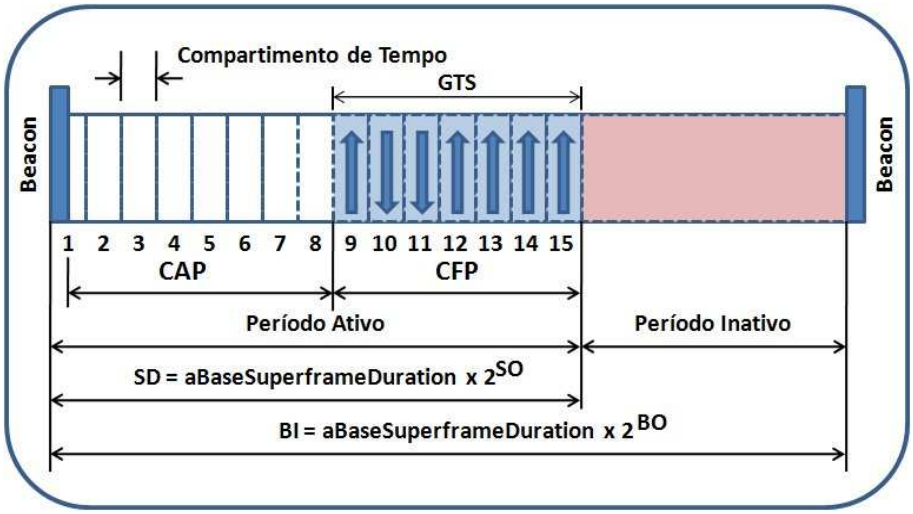


Figura 12 – Estrutura do superquadro IEEE 802.15.4™-2006 LR-WPAN

A estrutura do superquadro é definida por dois parâmetros: ordem do beacon, identificado por *Beacon Order* (BO) e ordem do superquadro, identificado por *Superframe Order* (SO), os quais determinam a duração temporal do superquadro e do período ativo, respectivamente. As durações temporais do superquadro e do período ativo são definidos como:

$$BI = aBaseSuperframeDuration * 2^{BO} \qquad \ldots\ldots (7)$$

$$SD = aBaseSuperframeDuration * 2^{SO} \qquad \ldots\ldots (8)$$

Nessas duas equações o termo "*aBaseSuperframeDuration*" é constante e expressa a máxima duração do superquadro quando BO é igual a zero. Na presente tese, na qual a Rede de Sensores Sem Fio opera em uma faixa de frequência de 2,4 GHz a uma taxa de comunicação de 250 Kbps essa constante corresponde a 960 símbolos, sendo que 1 símbolo corresponde a 4 bits, resultando em 15,36 ms de duração.

O compartimento de tempo garantido (GTS) é sempre alocado pelo coordenador da rede, seja por iniciativa própria, seja por meio de requisição dos nós sensores. Na situação em que o dispositivo interessado requer um compartimento de tempo garantido (GTS), o coordenador irá verificar se o recurso está disponível, e sendo possível atender a solicitação, aloca um GTS ao solicitante. Cada superquadro, individual, suporta até sete GTS e cada um deles deve conter um ou mais

compartimentos de tempo, sendo que a alocação do GTS não pode ultrapassar a duração mínima do sub-período CAP, o qual é fixado pela constante *aMinCAPLenght*, que equivale a 440 símbolos, resultando em um valor de 7,04 ms. Uma observação importante a ser destacada é que o dispositivo, mesmo que tenha um compartimento de tempo garantido alocado, ele também pode transmitir durante o sub-período de contenção CAP, ou seja, para um mesmo dispositivo é permitido transmitir nos dois sub-períodos, ou seja, no sub-período de contenção CAP e no sub-período livre de contenção CFP.

O período inativo, opcional, é utilizado no sentido de prover um mecanismo que possibilita a economia de energia, pois nesse período o dispositivo pode operar no modo de baixo consumo de energia, no qual ocorre o desligamento de várias funcionalidades dos dispositivos, entre elas o transceptor de rádio.

Em síntese a camada de controle de acesso ao meio é responsável pelas seguintes atividades:

- Gerenciar a emissão do sinal de beacon, caso o dispositivo seja o coordenador de rede;

- Prover acesso ao canal;

- Gerenciamento dos compartimentos de tempo garantidos (GTS);

- Validar os quadros descritos;

- Controlar a associação e desassociação de dispositivos na Rede de Sensores Sem Fio;

- Fornecer meios para implantar mecanismos de segurança.

3.3 PADRÃO ZIGBEE™

O padrão ZigBee™ foi desenvolvido por um consórcio de indústrias com interesses em comum, denominado de ZigBee Alliance. Esse padrão tem o objetivo de proporcionar suporte ao desenvolvimento de redes completamente conectadas com mais de 64.000 dispositivos sem fio cada uma. O conceito do ZigBee™ baseia-se na simplicidade e seu fundamento primário é a criação de um modelo que se torne padrão de uso para soluções de comunicação sem fio de baixo custo (GRATTON, 2007; ZIGBEE, 2008).

O padrão ZigBee™ fornece diretrizes para que exista a interoperabilidade entre os dispositivos desenvolvidos por empresas diferentes, por exemplo, um sensor que utiliza um nó padrão ZigBee™ de um determinado fabricante pode se comunicar com outro nó ZigBee™ de outro fabricante e estarem agregados na mesma rede, sem problemas de compatibilidade.

A aliança ZigBee™ se propõe a criar padrões inovadores onde projetos customizados são desenvolvidos por especialistas para atender a necessidade do mercado, tanto de fornecedores como de consumidores, facilitando as atividades diárias de controle.

Um dos papéis básicos da aliança ZigBee™ é realizar testes de homologação em produtos no sentido de verificar a aderência dos mesmos aos padrões. O objetivo é certificá-los, atestando suas funcionalidades. O consórcio ZigBee™ classifica os produtos em classes de aplicações, e atualmente abrange as seguintes áreas: automação predial, controle remoto, energia inteligente, cuidados com a saúde, automação residencial, dispositivos de entrada, serviços de automação comercial, serviços de telecomunicações e dispositivos para visualização de imagens tridimensionais.

O padrão ZigBee™ complementa o padrão 802.15.4 adicionando as camadas de rede (NWK), serviços de segurança de dados e uma subcamada responsável pela

interface com a aplicação, proporcionando um modelo de camadas de protocolo completo, indo da camada de aplicação até a camada física.

A figura 5 relaciona as camadas de protocolos constituintes de uma Rede de Sensores Sem Fio, nas quais a aliança ZigBee™ especifica as camadas 3 e 4. A figura 13 mostra em maiores detalhes os elementos constituintes dessas duas camadas.

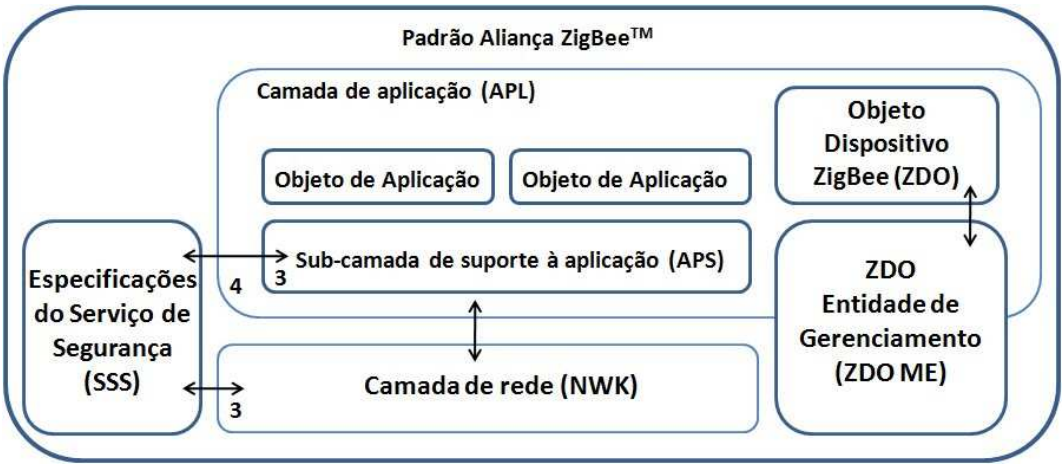


Figura 13 – Camadas de protocolos Aliança ZigBee™

A camada de rede é responsável, basicamente, pela definição da topologia da rede e também pelo roteamento dos dados na rede, por meio de multi-saltos, descobrimento e manutenção de rotas, segurança de dados, entrada e saída de dispositivos na rede. Essas funcionalidades são detalhadas a seguir.

3.3.1 Topologia de rede

No que se refere à topologia de rede o padrão ZigBee™ define três tipos de dispositivos constituintes de uma rede, o dispositivo terminal que corresponde ao elemento RFD do padrão 802.15.4, o dispositivo roteador ZigBee™ que corresponde ao elemento FFD com capacidade especial de roteamento e o terceiro dispositivo, também correspondente a um dispositivo FFD, denominado de coordenador ZigBee™, sendo que existe apenas um por rede.

O padrão ZigBee™ proporciona a criação de topologias de redes mais complexas do que as existentes no padrão 802.15.4. Além dos dois tipos de topologias de redes relacionadas no padrão 802.15.4, a topologia em estrela e a topologia ponto-a-ponto (também conhecida como topologia em malha), o padrão ZigBee™ possibilita a criação de redes hierárquicas denominada de topologia em árvore. Na figura 14 são apresentados exemplos dessas três topologias (BARONTI et al., 2007).

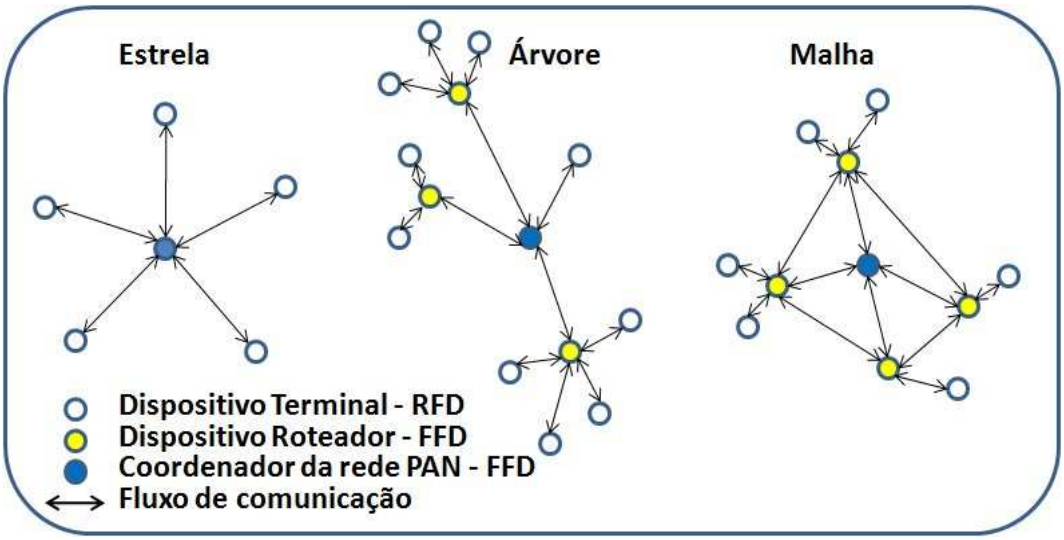


Figura 14 – Topologias de rede suportadas pelo padrão ZigBee™

A topologia em estrela oferece a opção de uma rede local pessoal, na qual apenas o coordenador ZigBee™ é responsável por toda a comunicação. Já a topologia em malha proporciona, para a aplicação, o papel do dispositivo terminal com características completas (FFD) ou reduzidas (RFD). Finalmente, na topologia em árvore, gerada a partir de múltiplas redes com topologia em estrela, possibilita a interligação de uma quantidade expressiva de nós.

3.3.2 Roteamentos suportados

Há dois esquemas de roteamentos diferentes suportados pelo padrão ZigBee™. O padrão não utiliza o conceito do protocolo da camada de transporte do modelo de referência OSI, que assegura a confiabilidade da comunicação ponto-a-ponto,

especificamente não existe a confirmação da comunicação via multi-saltos, apenas existe a garantia de comunicação em uma única conexão, ou seja, em um único salto, disponível a partir da camada de acesso ao meio (MAC) suportada pelo padrão 802.15.4.

O primeiro esquema de roteamento é denominado de roteamento em árvore, no qual todos os pacotes de dados trafegam pelas arestas da rede. O mecanismo funciona da seguinte maneira, um dispositivo terminal gera um pacote de dados e o envia para o roteador ou coordenador ZigBee™ ao qual ele está associado. Se o endereço de destino estiver associado a um nó filho ao mesmo roteador ou coordenador ZigBee™, o pacote de dados é enviado de forma descendente para esse nó. Caso contrário, o pacote de dados é enviado de maneira ascendente ao coordenador ZigBee™, o qual inicia o envio do pacote de dados de forma descendente na rede utilizando-se do ramo correto, o qual pode ser determinado a partir do endereço de destino, pois os endereços alocados aos dispositivos roteadores e terminais são acoplados com a topologia em árvore.

Esse tipo de topologia é uma boa escolha para redes nas quais os tráfegos de pacotes são direcionados de forma ascendente ao nó sorvedouro da rede. Outro ponto a se destacar, é que o roteador ZigBee™ não necessita de manter em memória uma tabela de roteamento completa, necessitando apenas conhecer seu roteador pai e os roteadores filhos, bem como os intervalos de roteamento (WILLIG, 2008).

O segundo esquema de roteamento, denominado de variante ZigBee™ AODV, é indicado para operar em redes com topologia em malha e é baseado no protocolo de roteamento denominado de *Ad-hoc On-demand Distance Vector* (AODV) (PERKINS; ROYER; DAS, 2003). Esse tipo de roteamento melhora o desempenho de redes ponto-a-ponto, nas quais, nós terminais em diferentes ramos, porém dentro do alcance de rádio mútuo, desejam se comunicar.

Com o protocolo de roteamento em árvore os pacotes teriam que trafegar através do coordenador ZigBee™, o que não é eficiente. O roteamento AODV é do tipo reativo, pois as rotas somente são calculadas quando elas são necessárias pela primeira

vez, ou seja, sob demanda. Este tipo de roteamento é diferente dos protocolos pró-ativos ou baseados em tabelas de rotas, no qual todos os roteadores mantêm de forma constante tabelas de roteamento atualizadas, mesmo em situações em que não existem pacotes de dados para serem roteados.

Os protocolos reativos são preferidos em situações em que as comunicações ocorram raramente, apresentando uma melhor eficiência em termos de consumo de energia, pois evita a manutenção constante das tabelas de roteamento de forma desnecessária. O ponto de destaque é que o levantamento das rotas disponíveis sob demanda, incrementa a latência na transmissão dos pacotes. No protocolo variante ZigBee™ AODV, um nó intermediário envolvido em uma rota ativa, mantém suas entradas na tabela de roteamento de forma persistente. Com o objetivo de lidar com as falhas na topologia da rede, em função da falha de um nó intermediário, o protocolo inclui mecanismos de manutenção e reparo de rotas. Esse tipo de protocolo é mais indicado para redes estáticas, com poucas alterações da topologia ao longo do tempo.

3.3.3 Suporte à segurança de dados

O padrão ZigBee™ fornece mecanismos especiais voltado para a questão da segurança dos dados. Segundo Gratton (2007) e Yang (2009) este padrão apresenta um nível de segurança de dados alto, pois adota um modo de operação que utiliza um padrão avançado de criptografia denominado de *advanced encryption standard* (AES) com até 128 bit de comprimento da chave criptográfica.

O mecanismo de segurança do padrão ZigBee™ é suportado por um modo de operação especializado denominado de *Counter with CBC-MAC* (CCM*), o qual engloba as funcionalidades do modo CCM padrão e adicionalmente oferece as funcionalidades de criptografia e integridade de forma independente. Vários níveis de segurança podem ser configurados no padrão ZigBee™, sendo utilizada a cifra de bloco conhecida como *Counter* (CTR) para o quesito de confidencialidade e a técnica conhecida como *Cipher Block Chaining Message Authentication Code* (CBC-

MAC) para o quesito integridade, e a combinação das duas técnicas para suportar confidencialidade e integridade de forma simultânea.

O quadro 4 relaciona os níveis de segurança suportados pelo padrão ZigBee™ em complexidade crescente. Na tabela o termo MIC simboliza "*Message Integrity Code*" e o termo CRI é a abreviação de criptografado.

Identificador do nível de segurança	**Tipo de segurança**	**Atributos de segurança**	**Criptografia dos dados**	**Integridade do quadro número bytes**
0x00	Sem	Sem	Sem	Sem(M=0)
0x01	AES-CBC-MAC-32	MIC-32	Sem	Com(M=4)
0x02	AES-CBC-MAC-64	MIC-64	Sem	Com(M=8)
0x03	AES-CBC-MAC-128	MIC-128	Sem	Com(M=16)
0x04	AES-CTR	CRI	Com	Sem(M=0)
0x05	AES-CCM-32	CRI-MIC-32	Com	Com(M=4)
0x06	AES-CCM-64	CRI-MIC-64	Com	Com(M=8)
0x07	AES-CCM-128	CRI-MIC-128	Com	Com(M=16)

Quadro 4 – Níveis de segurança disponíveis no padrão ZigBee™

No intuito de suportar esses algoritmos de forma eficiente, a maioria dos circuitos integrados que implementam o padrão ZigBee™ possuem um hardware dedicado ao processamento do padrão avançado de criptografia (AES). O padrão ZigBee™ apresenta algumas deficiências na segurança dos dados em função de interferências nos canais de comunicação, conflito de endereçamento, força da chave de criptografia e não-repúdio. Mesmo assim, o mecanismo de segurança adotado é considerado satisfatório em função das limitações de processamento, de memória e de consumo energético e também das expectativas de melhorias futuras no padrão ZigBee™.

3.3.4 Entrada e saída de dispositivos da rede

A camada de rede oferece serviços relacionados ao gerenciamento dos dispositivos da rede, por exemplo, configuração de um novo dispositivo, inicialização de uma nova rede, permissão da entrada, re-entrada e saída de um dispositivo da rede, associação de endereços aos dispositivos que se agregam a rede e descobrimentos de dispositivos vizinhos, entre outros serviços.

A camada de aplicação é o nível mais alto definido pelo padrão ZigBee™ e representa a interface entre o usuário e o sistema ZigBee™. A figura 13 detalha os componentes dessa camada, o que possibilita uma visão sistêmica. Os principais componentes desse nível são:

- Objeto dispositivo ZigBee™: Esse objeto, denominado de *ZigBee Device Object* (ZDO), é responsável pela definição de todos os dispositivos constituintes da rede, como coordenadores, roteadores ou terminais. Ele também é encarregado do descobrimento de novos dispositivos ingressantes na rede e da identificação dos serviços oferecidos, possibilitando o estabelecimento de uma conexão segura e de respostas às solicitações de participação da rede por dispositivos remotos de forma apropriada.

- Sub-camada de suporte a aplicação: O padrão ZigBee™ fornece uma sub-camada de suporte a aplicação denominada de *Application Support Sub-layer* (APS) que fica localizada no topo da camada de rede, fornecendo uma interface bem definida e também serviços de controle, ela trabalha como uma ponte entre a camada de rede e os componentes “objetos de aplicação”. A sub-camada APS fornece uma interface de até 240 pontos de interface com esses objetos, sendo que os pontos 0 e 255 são reservados para a interface de dados com o ZDO e para a difusão de dados para todos os outros pontos do sistema, respectivamente. O range de pontos de 241 até 254 está reservado para futuras implementações. O espaço de possíveis objetos de aplicações é estruturado em perfis, os quais tipicamente representam uma classe de aplicação, por exemplo, automação residencial, telecomunicações e

automação predial, entre outras. O conjunto de possíveis dispositivos com o mesmo perfil são agrupados em uma unidade denominada de cluster.

4 INSTRUMENTALIZAÇÃO DE SOFTWARE

Um instrumento tradicional é um equipamento ou mecanismo cuja principal função é fazer medições. Um multímetro é um dispositivo desenvolvido para possibilitar a medição de diversas grandezas elétricas, tais como resistência, corrente, tensão e capacitância, entre outras. Esse instrumento é projetado com uma finalidade específica de medir determinadas grandezas. O desenvolvimento da computação, na qual um hardware, controlado por um conjunto de instruções previamente sequenciadas, processa um conjunto de dados de entrada produzindo informações úteis na saída, ampliou a disponibilidade de instrumentos de medição.

Essa nova classe de instrumentos é conhecida como instrumento de software, criada a partir da combinação do computador e de programas dedicados. Esse dispositivo pode ser utilizado para medição, aquisição de dados, processamento de sinal, estimação, monitoração, detecção de falhas e diagnósticos (VOLOSENCU; MALITA, 2010).

4.1 CONCEITOS DE INSTRUMENTALIZAÇÃO DE SOFTWARE

O conceito de instrumentalização de software, segundo o dicionário padrão de termos elétricos e eletrônicos do IEEE (1984), é abordado de duas formas distintas, sendo a primeira definida como “Ferramentas de Instrumentalização” e a segunda como “Instrumentalização de Programa”. Segundo o dicionário citado a definição de ferramentas de instrumentalização é:

> “Uma ferramenta de software que gera e insere contadores ou outras sondas em pontos estratégicos de outro programa para fornecer estatísticas sobre a execução deste programa, tal como se o código do programa é executado completamente.” (IEEE, 1984, p.445, tradução nossa).

Para a instrumentalização de programa, o dicionário fornece duas definições:

a) “Sondas, tais como instruções ou declarações, inseridas no programa de computador para facilitar a monitoração da execução, prova de correção, monitoração de recursos ou outras atividades.”;
b) “O processo de preparar e inserir sondas em um programa de computador.” (IEEE, 1984, p.689, tradução nossa).

Das definições apresentadas pode-se notar o caráter amplo e abrangente de ferramentas de instrumentalização, no sentido de criar as condições necessárias para a avaliação do programa de interesse. Já a definição de instrumentalização de programa, possui caráter específico e restrito, ou seja, impõe a necessidade de inserção de código apropriado, diretamente do programa de interesse. Um ponto em comum nas duas visões é o objetivo final, que é a implantação da capacidade de medição de parâmetros de interesse, por meio de artifícios que interagem com o programa sob avaliação e proporcionam a mensuração de características desejadas, como por exemplo, o comportamento da utilização da memória, do processador, o uso de determinada instrução e a frequência e duração das chamadas de procedimentos, entre outras.

Atualmente, com a solidificação das bases da engenharia de software e a análise sistemática de ambientes computacionais, a avaliação de desempenho das aplicações tem um papel de destaque, proporcionando a utilização de ferramentas de análise de desempenho, de forma acentuada. O desenvolvimento na área de ferramentas de análise de desempenho por software surgiu no início dos anos 80, com a publicação em 1982 do artigo “Gprof: A Call Graph Execution Profiler”, escrito por Graham, Kessler e Mckusick (1982). Esse artigo serviu como base para o desenvolvimento de uma ferramenta para o ambiente UNIX, denominada de “Gprof”. Essa ferramenta cria um grafo detalhado de chamadas de funções de um programa, registrando a sequência e o tempo gasto em cada função (WOOD, 2008).

Posteriormente, mais precisamente em 1994, a área de instrumentalização de software ganhou a contribuição da *Digital Equipment Corporation* (DEC) que detalhou a operação de um conjunto de ferramentas denominado de *Analysis Tool with OM* (ATOM). Esta ferramenta é mais abrangente que o Gprof e possibilita o

desenvolvimento de rotinas de monitoração, característica que possibilita o desenvolvimento de instrumentalização mais abrangente, eliminando a limitação imposta pelo Gprof. O Gprof disponibiliza conjunto de serviços pré-definidos, como contadores ou rastreamento de posições de memória. O ATOM permite que chamadas de procedimentos sejam inseridas antes ou após qualquer programa, procedimento, bloco básico ou instrução, pois de acordo com os autores, um programa pode ser visualizado como uma coleção linear de procedimentos, que por sua vez é constituído por blocos básicos, que são constituídos por instruções.

A ferramenta ATOM é estruturada em dois passos, conforme pode ser verificado na figura 15. No primeiro passo os módulos objetos genéricos são combinados com as rotinas de instrumentalização desenvolvidas pelo usuário para a criação da ferramenta de instrumentalização personalizada. Essa ferramenta irá instrumentar o programa em pontos especificados pelas rotinas de instrumentalização. No segundo passo, essa ferramenta gerada é aplicada ao programa sob análise para produzir um programa executável instrumentalizado.

O programa executável instrumentalizado é organizado de tal forma que as informações do programa aplicativo comunicam-se diretamente com os procedimentos das rotinas de análise por meio de chamadas de procedimentos. Os dados são passados como argumentos, eliminando a necessidade da utilização de um mecanismo de despacho. No sentido de melhorar a eficiência de comunicação, o programa de aplicação e as rotinas de análise são executados no mesmo espaço de endereçamento de memória. Outra característica importante da ferramenta é que ela é independente de qualquer compilador e linguagem, porque ela opera sobre os módulos objetos. (SRIVASTAVA; EUSTACE, 1994).

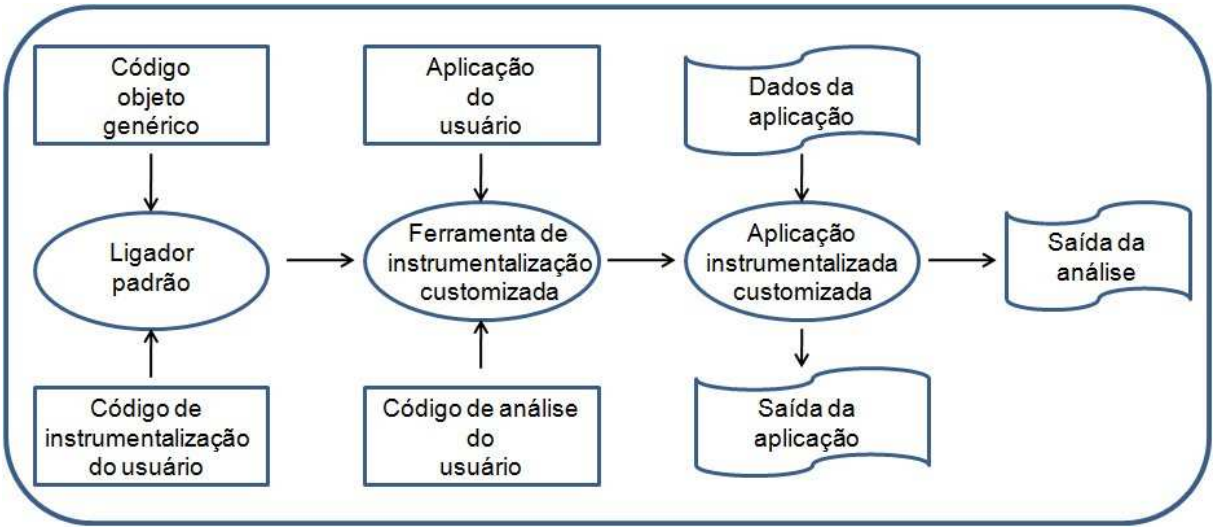


Figura 15 – Estrutura da ferramenta ATOM – Fonte: (SRIVASTAVA; EUSTACE, 1994)

É importante notar a utilização do termo “Profiler” no título do artigo de Graham, Kessler e Mckusick (1982), representando ferramentas de análise de programas que possibilitam uma melhor compreensão de seus comportamentos. Nesse contexto, “Software Profiling” é o ato de criar as condições necessárias para que seja executado o Profiler, cujas informações de saída fornecidas, conhecidas como “Profile”, permitem a análise do desempenho e comportamento do programa. Vários profissionais da área de computação utilizam a técnica de Profiling em suas atividades. Desenvolvedores de hardware utilizam essas ferramentas para analisar como determinados programas irão se comportar em arquiteturas computacionais diferentes. Desenvolvedores de software se beneficiam desse ambiente, por meio da análise de desempenho dos programas desenvolvidos no sentido de identificar trechos de códigos críticos. Desenvolvedores de compiladores conseguem avaliar as estratégias adotadas no sequenciamento de instruções e predição de desvios.

O termo “Profiling” representa uma técnica de exame de programas que inclui tracing (rastreamento) e debugging (depuração) em tempo de execução. Existe uma técnica similar ao Profiling denominada de Tracing, porém possui um foco diferente. O Tracing é utilizado para examinar a lógica e não o desempenho do programa sob avaliação. A técnica de tracing consiste em registrar informações de interesse para avaliar a correção da lógica do programa (LEVON, 2002).

4.2 PRINCIPAIS FORMAS DE IMPLANTAÇÃO

Geralmente a aplicação alvo da análise deve incluir algum nível de instrumentalização para suportar a operação dos mecanismos de Profiling. Segundo Levon (2002), existem algumas técnicas de instrumentalização em uso, em função das ferramentas de análise baseadas em totalizadores, destacando-se:

- **Simulação** – Um simulador pode fornecer dados detalhados como resultado de um processo de simulação. Essa técnica tende a ser obstrutiva e lenta. Segundo Magalhães e Pinheiro (2010) essa abordagem necessita da criação de um modelo que simule o funcionamento do sistema a ser avaliado, o qual deve representar com um grau desejado as características funcionais do sistema em uma escala de tempo adequada.

- **Instrumentalização no nível do código fonte** – Esta técnica consiste na alteração do código fonte da aplicação de interesse, por meio da inserção de trechos de código de profiling. Essa inserção pode ser feita de forma semi-automática por meio do pré-processador no estágio de compilação do programa fonte, ou manualmente, requerendo que o programador adicione explicitamente chamadas para os trechos de código de profiling, conhecidos como *Application Program Interface* (API).

- **Instrumentalização em tempo de compilação** – O próprio compilador pode ser utilizado para inserir o código de profiling. Esta técnica é mais conveniente que a anterior e também impõe a necessidade de nova compilação da aplicação sob análise.

- **Instrumentalização binária offline** – Este método tem implantação complexa e consiste em reescrever o código em imagens binárias no sentido de adicionar a instrumentalização desejada. Esse é um método complexo de ser implantado, porém apresenta um desempenho melhor em termos de consumo extra de recursos (capacidade de processamento e memória).

- **Instrumentalização binária online** – Técnica utilizada por compiladores em tempo de execução, a qual consiste em adicionar o nível de instrumentalização desejada no código binário gerado durante sua execução.

Vale destacar que apesar da importância da instrumentalização de software sempre ocorre uma sobrecarga no processamento original da Rede de Sensores Sem Fio. Dessa forma torna-se importante ter instrumento que possibilitem efetuar a medição do nível de interferência proporcionado pela instrumentação.

Uma pesquisa sobre técnicas de lançamento de Redes de Sensores Sem Fio em campo foi desenvolvida por Beutel et al. (2010), em virtude que, apesar dos esforços no desenvolvimento da tecnologia desse tipo de redes, os casos de sucesso no lançamento e utilização dessas redes, na prática, ainda são poucos. Os autores revisaram 14 diferentes projetos com o objetivo de entender melhor o porquê do insucesso no lançamento de Redes de Sensores Sem Fio. A pesquisa apresenta métodos e ferramentas de apoio na localização de falhas e que auxiliam no levantamento das causas raízes das mesmas. Essas técnicas de instrumentalização são especialmente projetadas ou adaptadas para análise de redes de sistemas embarcados distribuídos ao nível de componente, de nó sensor e de rede.

A instrumentalização dos nós sensores está relacionada com modificações nos nós sensores tanto em termos de software, como em termos de hardware para possibilitar acesso ao estado do sistema. Uma observação importante é feita no sentido de minimizar a interferência introduzida com a adição da instrumentação, conhecida como efeito sonda, que consiste na alteração do comportamento do sistema em virtude da instrumentalização.

A instrumentalização de software é necessária para recuperação do estado do software em execução no microprocessador. Felizmente não é necessário, na prática, acompanhar a evolução do estado do software continuamente, pois seria praticamente impossível transferir a quantidade de informação necessária devido à largura de banda limitada. Portanto a monitoração da alteração de algumas variáveis do programa em determinados momentos é suficiente para a maioria das análises.

De acordo com os autores, a instrumentalização de software pode ser realizada em dois níveis: a) no nível do programa fonte, no qual o código fonte do programa é modificado ou, b) no nível do código binário, no qual o código binário em linguagem de máquina é alterado. Outro aspecto a ser levado em consideração é o que deve ser instrumentalizado, se é o código da aplicação e/ou o sistema operacional.

Em alguns casos é suficiente apenas instrumentalizar o sistema operacional ou o ambiente de execução para registrar a temporização e parâmetros de chamadas do núcleo do sistema operacional, bem como a alocação de memória e o chaveamento de contexto.

Em sistemas, nos quais é utilizada uma máquina virtual para interpretar o bytecode (código binário que representa a instrução do programa de aplicação para a máquina virtual), a instrumentalização do interpretador possibilita uma visão detalhada do estado da aplicação sem a necessidade de se instrumentalizar o código fonte da aplicação. Apesar de que algumas máquinas virtuais tenham sido desenvolvidas para Redes de Sensores Sem Fio, elas ainda são pouco utilizadas devido à degradação de desempenho em função dos escassos recursos disponíveis.

Em algumas ocasiões é necessária a alteração do nível de instrumentalização durante a execução do programa, por exemplo, no sentido de se implantar um mecanismo de depuração interativo, no qual se deseja colocar pontos de verificação que são disparados automaticamente quando ocorre a alteração de alguma variável de interesse. Esse mecanismo é conhecido como instrumentalização dinâmica. Apesar de parecer atraente num primeiro instante, esse mecanismo apresenta desafios significantes para sua utilização na prática.

A revisão, em campo, de uma instrumentalização já efetuada, é extremamente não recomendável, uma vez que a mesma precisa ser novamente carregado na memória dos microcontroladores dos nós sensores e do sorvedouro. A instrumentalização do código binário é possível, porém o custo em termos de consumo de energia na substituição do programa executável na memória flash do microcontrolador é proibitivo, e também existe uma limitação no número de gravações da memória flash, embora seja na ordem de milhares de vezes. Esses desafios podem ser

minimizados com a utilização de máquina virtual, uma vez que somente o interpretador fica armazenado na memória flash e o programa a ser interpretado reside na memória RAM.

4.2.1 Métodos de Instrumentalização de Software em Redes de Sensores Sem Fio

Vários métodos foram desenvolvidos, ao longo do tempo, no sentido de auxiliar o desenvolvimento, os testes e o lançamento de sistemas completos de Redes de Sensores Sem Fio. Os nós sensores requerem uma instrumentação de software cuidadosa e orquestrada com o objetivo de possibilitar que eles interajam com todos os dispositivos da rede. Nesse sentido, instrumentalização de rede refere-se a instrumentalização dos nós sensores com o intuito de extrair informações que possibilitem uma visão ampliada da rede de sensores (BEUTEL et al., 2010).

A seguir são destacados cinco métodos diferentes de instrumentalização de redes: coleta por meio da própria rede; armazenamento local; monitor offline; monitor online; e coleta por meio de rede adicional. A figura 16 ilustra esses métodos. Nessa figura a Rede de Sensores Sem Fio é destacada por meio de um retângulo tracejado, os nós monitores são representados por um quadrado, os dispositivos de armazenamento por um disco, o nó sorvedouro por um microcomputador, a comunicação sem fio por linhas tracejadas e a comunicação por fio em linha cheia.

Coleta por meio da própria rede – *In-band Collection* – Neste método as mensagens de monitoração são encaminhadas através da Rede de Sensores Sem Fio até o nó sorvedouro, o qual se encarrega de processar as mensagens e alimentar o sistema de avaliação. Essa abordagem não requer a utilização de hardware adicional, mas ocasiona alteração na aplicação e também provoca aumento no tráfego de mensagens da rede. Esse método é ilustrado pela figura 16(a).

Armazenamento local – *logging* – A principal característica desta abordagem é o armazenamento local das mensagens de monitoração, conforme pode ser verificado na figura 16(b). Esses dados são armazenados na memória flash dos nós sensores

para posterior coleta. Apesar desse método não interferir com o tráfego de dados da aplicação, existe a necessidade de se acessar fisicamente os nós para a coleta das mensagens armazenadas, operação que resulta em um trabalho adicional e muitas vezes inviável na prática. Outro desafio a ser levado em consideração é a capacidade de armazenamento da memória flash disponível.

Monitor offline – *offline sniffer* – Neste método um conjunto de nós monitores adicionais é adicionado à Rede de Sensores Sem Fio com o objetivo de capturar as mensagens enviadas pelos nós sensores. Esses dados são armazenados na memória flash dos nós monitores, conforme destacado na figura 16(c). De forma similar ao mecanismo anterior, é necessário o acesso físico ao nó monitor para a coleta das mensagens armazenadas. Esse mecanismo é totalmente passivo e não interfere na aplicação como também não interfere na rede. Outra possibilidade é alterar o software do nó sensor para que envie mensagens de monitoração para os nós monitores, de forma que os outros nós sensores não propaguem essas mensagens, reduzindo de forma significante o tráfego na Rede de Sensores Sem Fio.

Monitor online – *online sniffer* – Esta forma de instrumentalização de Redes de Sensores Sem Fio, também utiliza de um conjunto de nós monitores para capturar as mensagens da rede de forma similar ao mecanismo método descrito anteriormente, porém a diferença básica é que esses nós monitores possuem um segundo canal de comunicação via rádio, geralmente mais potente e com maior alcance, por exemplo suportado por Bluetooth ou por *Wireless Local Area Network* (WLAN), através do qual os nós monitores enviam as mensagens capturadas, para um nó sorvedouro de monitoração sem a necessidade da coleta manual das mensagens. A figura 16(d) destaca o canal secundário em linha tracejada e ponto em vermelho. Esse método deve possuir o segundo canal de comunicação de rádio confiável. A vantagem dessa abordagem é a possibilidade do processamento online das mensagens monitoradas.

Coleta por meio de rede adicional – *out-of-band Collection* – Este método não utiliza a comunicação sem fio para o propósito de monitoração de mensagens, dessa forma não existe nenhuma possibilidade de interferência com a Rede de Sensores

Sem Fio. Nesse sentido é construído um segundo canal de comunicação suportado por cabos. Dependendo da forma de encaminhamento das mensagens de monitoração para o nó sorvedouro, esse mecanismo pode ser classificado em: (a) monitoração sem fio, no qual as mensagens de monitoração dos nós sensores são enviadas por meio de fio, em geral por um canal serial, ao nó monitor que por sua vez se comunica com o nó sorvedouro por um canal sem fio. A figura 16 (e) ilustra esse tipo de monitoração sem fio; e a figura 16 (f) ilustra a monitoração por meio de fio, no qual os nós sensores enviam diretamente as mensagens de monitoração ao nó sorvedouro por meio de fios.

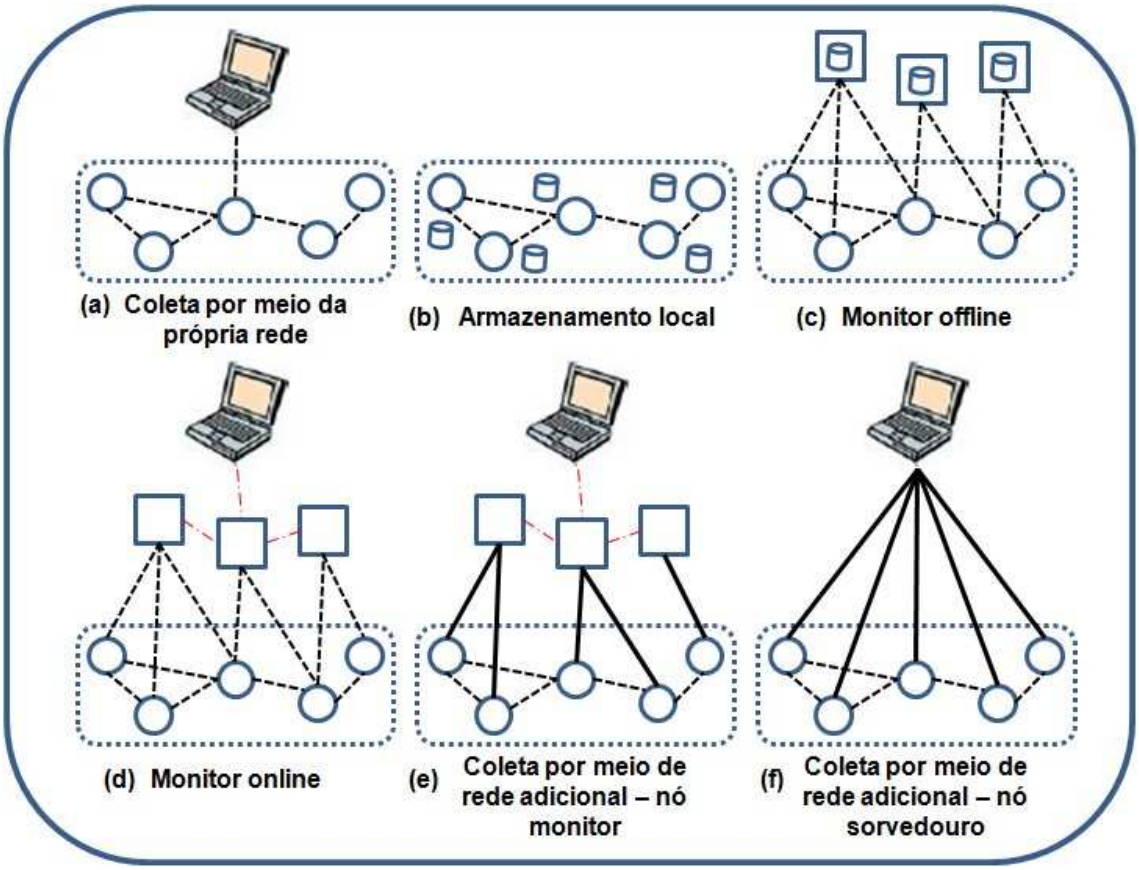


Figura 16 – Possíveis métodos de instrumentalização de uma Rede de Sensores Sem Fio

O quadro 5 relaciona os métodos utilizados na instrumentalização de Redes de Sensores Sem Fio, destacando as seguintes características: a) conexão física; b) volume de dados; c) monitoramento online; e, d) identificação dos possíveis métodos de instrumentalização de uma RSSF de acordo com a figura 16.

Método	Conexão física	Volume de dados	Monitoramento online	Figura 16
Coleta por meio da própria rede	Não	Baixo	Sim	(a)
Armazenamento local	Não	Baixo	Não	(b)
Monitor offline	Não	Baixo	Não	(c)
Monitor online	Não	Baixo	Sim	(d)
Coleta por meio de rede adicional	Sim	Alto	Sim	(e),(f)

Quadro 5 – Resumo das características dos mecanismos de instrumentalização

4.3 APLICAÇÕES

O conceito de instrumentalização de software é amplo e tem sido utilizado de forma crescente, acompanhando o desenvolvimento de sistemas cada vez mais complexos. Atualmente é inviável a monitoração de sistemas computacionais complexos e distribuídos por meio da utilização de instrumentos tradicionais, sendo, portanto necessário o desenvolvimento de instrumentos, suportados por software.

Em Redes de Sensores Sem Fio a abordagem de sua monitoração por meio de instrumentalização de software é atrativa e necessária devido a suas características de recursos limitados, da quantidade de nós sensores, da acessibilidade física aos elementos, da instabilidade da topologia da rede e da imprevisível interação do meio físico com os nós sensores.

Vários trabalhos foram desenvolvidos com esse conceito. A seguir destacam-se alguns desses trabalhos.

O trabalho de Volosencu e Malita (2010) apresenta uma solução técnica para a monitoração de Redes de Sensores Sem Fio utilizando uma ferramenta comercial de mercado, da empresa americana *National Instruments* (NI), conhecida como LabView. Essa ferramenta possibilita a criação de instrumentos de software (programa de instrumentalização) de forma rápida e relativamente simples. Segundo os autores, uma Rede de Sensores Sem Fio pode ser considerada como um sensor

distribuído que é utilizado para medir grandezas físicas como temperatura, aceleração, pressão, luminosidade e umidade, entre outras grandezas.

Nesse trabalho, os nós sensores são compostos por dois módulos, sendo um módulo de processamento e outro módulo de sensoriamento, que permite a medição das seguintes grandezas físicas: temperatura, umidade, pressão, iluminação e aceleração. Os nós sensores se comunicam com a estação base, constituída por um módulo de processamento, com funcionalidade de comunicação *Universal Serial Bus* (USB), e um módulo para a comunicação de dados com os nós sensores da Rede de Sensores Sem Fio.
O instrumento de software é um programa executado em um equipamento com capacidade computacional considerável e desempenha as seguintes funções: inicialização, encerramento, aquisição de dados enviados pelos nós sensores, visualização dos dados de diversas maneiras, estimação e registro dos dados em uma base de dados. O trabalho praticamente não aborda as características da rede utilizada e também não define as condições de contorno, como número de nós sensores utilizados, arquitetura utilizada, quais foram os testes realizados, ambiente utilizado e outras informações importantes.

Hammad e Cook (2009) desenvolveram uma ferramenta de instrumentalização, denominada de “Limow”, para ser utilizada em conjunção com o TinyOS, um sistema operacional projetado para dispositivos embarcados com comunicação sem fio de baixa potência, como os utilizados em Redes de Sensores Sem Fio. O TinyOS é referência em sistemas operacionais para essa classe de redes.

Os autores relatam que simuladores são importantes no desenvolvimento de Redes de Sensores Sem Fio, porém apresentam algumas deficiências, entre elas a dificuldade de se modelar o ambiente real. A monitoração em campo por software fornece capacidades adicionais que auxiliam ao desenvolvedor ter uma melhor compreensão do sistema que está sendo projetado ou instalado, e adicionalmente, se o sistema está operando conforme concebido.

O objetivo principal da ferramenta é instrumentalizar o software dos nós sensores, para que localmente coletem dados e periodicamente enviem esses dados para um

ponto central, denominado de *gateway* para um armazenamento permanente e para inspeção. A instrumentalização é realizada utilizando-se as características nativas do ambiente de desenvolvimento do TinyOS. Os programas a serem executados com o suporte do TinyOS são desenvolvidos utilizando a linguagem de programação conhecida como “nesC”, que é mais amigável para o desenvolvimento de sistemas a eventos discretos, facilitando a criação de aplicações para esse tipo de redes.

Um pré-processador converte o programa escrito nessa linguagem para a linguagem “C”, que finalmente é compilada para o código executável do processador alvo. Existe uma opção de compilação que possibilita a inserção de trechos de códigos para monitoração do programa, ou seja, realizar o “Profiling” do programa. Os autores aproveitam essa funcionalidade e sobrescrevem o código original de profiling com o “instrumento” desejado, o qual captura e registra a ocorrência e duração de cada chamada de função e seu retorno. No desenvolvimento da ferramenta foi observado o cuidado de se implantar um mecanismo de controle da quantidade de instrumentalização adicionada, pois originalmente o compilador gera pontos de monitoração para todas as funções do programa.

Segundo os autores o custo, em termos de acréscimo de processamento, pode ser proibitivo para o hardware alvo. Outro ponto de destaque é que a ferramenta adiciona instrumentalização sem adicionar nenhuma modificação no programa fonte. Com os dados armazenados em uma estação base é possível gerar informações úteis ao desenvolvedor. A ferramenta fornece duas possíveis visões: a) o comportamento temporal detalhado do programa e b) um resumo do comportamento temporal do programa.

5 PROPOSTA DE INSTRUMENTALIZAÇÃO PARA REDES DE SENSORES SEM FIO

O objetivo do presente capítulo é descrever a proposta da utilização do conceito de instrumentalização de software para o levantamento de perfil (*profile*), em Redes de Sensores Sem Fio suportadas pelo padrão IEEE 802.15.4^{TM}-2006 LR-WPAN, em aplicações com restrições temporais, como, por exemplo, em aplicações industriais, nas quais esse tipo de rede pode ser utilizado como um sistema redundante de controle, ou até mesmo, como o próprio sistema de controle, desde que observadas as limitações intrínsecas dessa tecnologia. Os perfis levantados são: aspectos temporais e manutenção da qualidade da conexão.

5.1 CONSIDERAÇÕES SOBRE A PROPOSTA E CONTRIBUIÇÕES PRINCIPAIS

O foco da instrumentalização proposta neste trabalho é bastante amplo, não se restringindo a aspectos particulares das Redes de Sensores Sem Fio. Desta forma, seu foco principal é a proposta de instrumentalização de redes suportadas pelo padrão 802.15.4^{TM}-2006 LR-WPAN, utilizadas em aplicações que tenham restrições e aderência a requisitos temporais, por meio de software, que possibilite o levantamento do perfil dos aspectos temporais e da manutenção da qualidade da conexão da rede, auxiliando o usuário a realizar a monitoração automática e avaliação do sistema.

O conceito de instrumentalização de software aplicado a Redes de Sensores Sem Fio é implantado por meio da inserção de informações especiais nas mensagens trocadas na rede, seguida pela consolidação dessas informações pelo instrumento de software. Os benefícios da instrumentalização da rede levam, necessariamente, a um aumento de tráfego e consumo de recursos da rede e devem ser considerados no estágio de desenvolvimento do projeto.

A instrumentalização permite a criação de ferramentas para o estudo e avaliação de Redes de Sensores Sem Fio de forma mais ampla, não ficando restrita à influência de um único fator (por exemplo: vida útil da bateria), permitindo a visualização de sua relação com os demais fatores (por exemplo tempo de ida-e-volta, energia detectada, indicador da qualidade da conexão e temperatura), bem como da influência do ambiente sobre a rede.

O presente trabalho contribui para o conhecimento ampliado da rede, o que proporciona a abordagem dos desafios descritos na seção 2.3 de uma maneira mais eficiente. Por exemplo, em relação à vida útil, a instrumentalização possibilita o acompanhamento do consumo de energia por um determinado nó sensor, e dessa forma permite estimar a sua vida útil, ou até mesmo a vida útil da rede como um todo. No quesito qualidade de serviço, essa abordagem proporciona a avaliação da Rede de Sensor Sem Fio como um sistema inserido no ambiente, avaliando a manutenção da qualidade da conexão e os atrasos temporais envolvidos e não apenas o resultado da taxa de erro de bit (BER). Os desafios impostos devido aos aspectos temporais também podem ser instrumentalizados, possibilitando a mensuração dos tempos envolvidos, que podem ser categorizados como atrasos devido ao envio, processamento e recebimento de mensagens. Com essas informações o projetista da rede, pode verificar quais são os pontos críticos do projeto.

5.2 INSTRUMENTALIZAÇÃO DE REDES DE SENSORES SEM FIO SUPORTADA PELO PADRÃO IEEE 802.15.4™ – 2006 LR-WPAN

Foi selecionada a técnica de alteração do código fonte com o intuito de demonstrar a instrumentalização de aplicações já desenvolvidas e que o conceito pode ser utilizado em vários tipos de aplicações. As outras técnicas mencionadas, como instrumentalização em tempo de compilação, binária offline e binária online exigem o desenvolvimento de um ambiente adicional que foge ao escopo do presente trabalho.

O presente trabalho se propõe a instrumentalizar Redes de Sensores Sem Fio suportadas pelo padrão IEEE 802.15.4™-2006 LR-WPAN, utilizadas em aplicações com restrições temporais, por meio da alteração dos códigos fontes dos nós sensores e do nó sorvedouro e o desenvolvimento de um instrumento de software. O objetivo é o levantamento do perfil da rede em termos dos aspectos temporais e da manutenção da qualidade da conexão. Na figura 17 é possível observar os componentes típicos de uma Rede de Sensores sem Fio instrumentalizada e que serve de base para a proposta de instrumentalização desse tipo de rede.

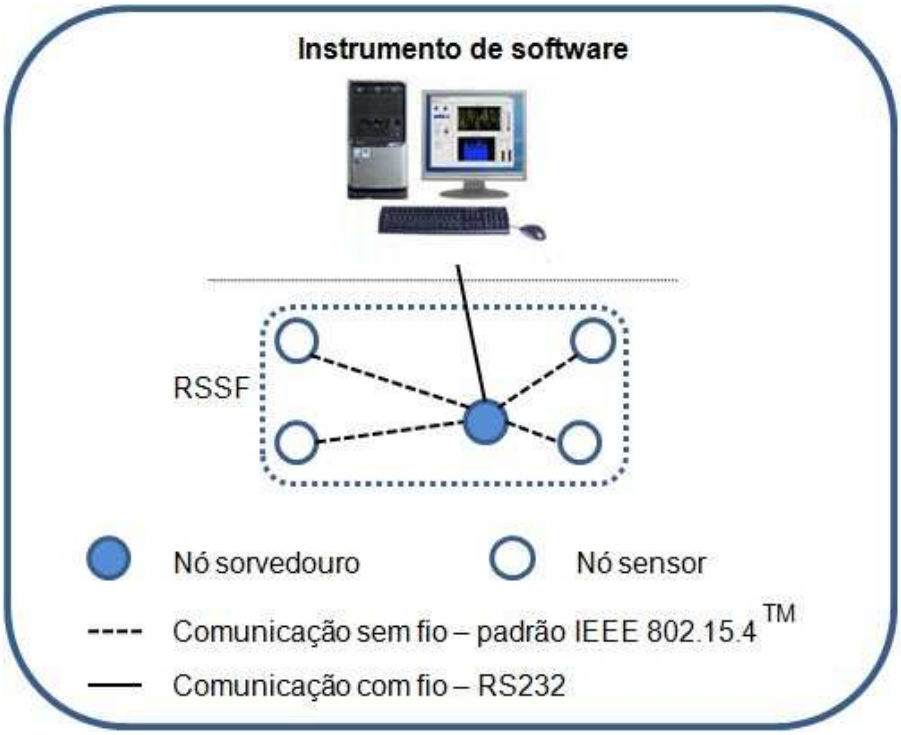


Figura 17 – Componentes típicos de uma Rede de Sensores Sem Fio instrumentalizada

5.2.1 Instrumentalização do nó sensor e do nó sorvedouro

A adoção do padrão IEEE 802.15.4™-2006 LR-WPAN facilita em parte o processo de instrumentalização da rede, pois esse padrão especifica a disponibilidade de algumas métricas de interesse, como por exemplo, a energia detectada, o indicador da qualidade da conexão e a taxa de erro de pacotes, entre outras.

A instrumentalização do nó sensor consiste na criação de uma mensagem sonda que contenha informações de interesse para o levantamento do perfil e a análise do comportamento da rede. A figura 18 ilustra o quadro de dados da camada de acesso ao meio (MAC) identificado por MPDU, e também o pacote de dados a ser transmitido pela camada física (PHY), identificado por PPDU. Na figura a região em destaque, identificada como “Dados/sonda”, é destinada à inserção das métricas desejadas. Vale observar que o quadro MPDU completo da camada de acesso ao meio (MAC) é a carga útil do pacote da camada física (PHY), identificado na figura como PSDU. Como resultado, tem-se um pacote com informações de métricas de interesse denominado doravante de “pacote sonda”.

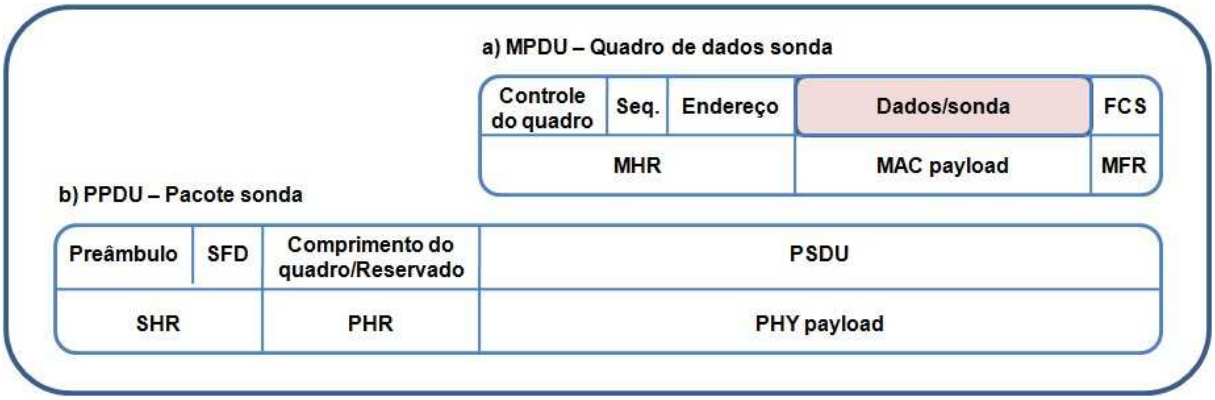


Figura 18 – Estrutura do pacote sonda

Como as Redes de Sensores em Fio possuem uma ampla gama de aplicações, existe um grande número de métricas de interesse que dependem da aplicação. O presente trabalho engloba parâmetros relacionados com restrições temporais e com a manutenção da qualidade da conexão da rede. Dessa forma o conceito introduzido pela presente tese pode ser utilizado como base para a avaliação dessa classe de aplicações. Com o intuito de comprovar a viabilidade de instrumentalização de software de Redes de Sensores Sem Fio, foi criado um nível de instrumentalização no nó sensor e no nó sorvedouro que seja minimamente intrusivo e que forneça uma visão consistente da rede.

O parâmetro comum à classe de aplicações com restrições temporais, e que modela os aspectos temporais da rede é: tempo de ida-e-volta – *Round Trip Time* (RTT).

Os parâmetros relacionados com a operação da rede, responsáveis pelo conhecimento do nível da manutenção da qualidade das conexões da rede, são: energia detectada; indicador da qualidade da conexão; taxa de erro de pacote; taxa de erro de pacote recebido; tensão da bateria e temperatura de operação.

Os parâmetros citados foram selecionados em virtude dos mesmos serem independentes do tipo de aplicação, e que se propõem a auxiliar no conhecimento do comportamento da rede em termos temporais e em termos de manutenção da qualidade da conexão.

O quadro 6 apresenta uma síntese dos parâmetros de interesse utilizados nesta tese, adicionando-se a informação se os mesmos são ou não especificados pelo padrão IEEE 802.154^{TM}-2006 LR-WPAN. Vale a pena destacar que no caso dos parâmetros não especificados, os mesmos foram incluídos em virtude de uma análise que os apontou como sendo de importância comum as aplicações de Redes de Sensores Sem Fio.

	Métricas Instrumentalizadas			
	Seq.	**Sigla**	**Significado**	**Padrão**
Aspectos Temporais	1	RTT	Tempo de ida-e-volta	Não
Manutenção da qualidade da conexão	1	ED	Energia detectada	Sim
	2	LQI	Indicador da qualidade da conexão	Sim
	3	PRR	Taxa de pacote recebido	Não
	4	TB	Tensão da bateria	Não
	5	TO	Temperatura de operação	Não

Quadro 6 – Métricas Instrumentalizadas

Para que a construção do pacote sonda do nó sensor suporte a monitoração e avaliação das métricas relacionadas é necessária a inclusão das seguintes

informações no campo MAC payload do quadro dados da camada de acesso ao meio (MPDU):

- Código da operação (CodOp) – Responsável pela especificação da natureza da operação. É um campo de 1 byte. No presente trabalho o código de operação 0x01 representa a avaliação da análise temporal, que consiste no envio do pacote sonda pelo nó sorvedouro e o retorno imediato desse pacote recebido pelo nó sensor ao nó de origem. O código de operação 0x02 representa a avaliação da manutenção da qualidade, que consiste no envio de pacotes sonda pelo nó sensor ao nó sorvedouro. A quantidade de pacotes e o intervalo de envio são especificados no pacote sonda do nó sorvedouro.

- Endereço do nó sensor (EndNsr) – Campo que contém o endereço do nó sensor no padrão especificado pelo padrão IEEE 802.15.4™-2006 LR-WPAN, composto por 8 bytes.

- Relógio no instante da transmissão do nó sorvedouro (RelTxSrv) – Representa o valor do relógio do nó sorvedouro no momento do envio do pacote sonda. Esse valor é utilizado pelo nó sorvedouro para cálculo da latência total entre o envio do pacote sonda e o recebimento da resposta.

- Relógio do nó sensor no instante da recepção (RelRxNsr) – Representa o valor do relógio do nó sensor no instante da recepção de um pacote. Esse valor é utilizado para o cálculo da latência entre a chegada do pacote sonda do nó sorvedouro e o envio do pacote sonda do nó sensor como resposta.

- Relógio do nó sensor no instante da transmissão (RelTxNsr) – Representa o valor do relógio do nó sensor no instante da transmissão de um pacote. Esse valor é utilizado para o cálculo da latência entre a chegada do pacote sonda do nó sorvedouro e o envio do pacote sonda do nó sensor como resposta.

- Tensão da bateria (TenBatNsr) – Representa o valor da tensão da bateria do nó sensor em milivolts. Utiliza um campo de dois bytes no pacote sonda.

- Valor da temperatura (TempNsr) – Representa o valor da temperatura de operação do nó sensor. O campo utiliza dois bytes no pacote sonda e é expresso em graus Celsius.

- Indicador da qualidade da conexão na recepção (LqiNsr) – Valor representativo da qualidade da conexão, utiliza 1 byte, sendo que valores baixos representam baixa qualidade da conexão e valores altos representam alta qualidade da conexão.

- Energia detectada na recepção (EDNsr) – Representa a energia recebida pelo nó sensor no momento da recepção de um pacote. Valor utilizado em conjunto com o valor Lqi para avaliar a manutenção da qualidade da conexão. Esse campo ocupa 1 byte e expressa a energia detectada em dBm.

- Número sequencial de pacotes transmitidos pelo nó sorvedouro (NumSeqSrv) – Identifica a sequência de pacotes transmitidos pelo nó sorvedouro, a cada pacote sonda enviado é inserido um campo com um número sequencial de 2 bytes. Esse campo é utilizado para avaliar perdas na recepção de pacotes pelo nó sensor e pelo próprio nó sorvedouro. Esse valor é uma cópia do valor sequencial do pacote sonda do nó sorvedouro.

- Número sequencial de pacotes transmitidos pelo nó sensor (NumSeqNsr) – Da mesma forma do campo anterior o nó sensor também adiciona um número sequencial a cada pacote transmitido, possibilitando, dessa forma, a avaliação da perda de algum pacote pelo nó sorvedouro. Esse campo ocupa dois bytes.

A estrutura da carga útil do quadro de dados sonda do nó sensor, inserida no campo MAC payload, pode ser visualizada na figura 19, na qual está relacionado o sequenciamento das informações descritas anteriormente, bem como a largura em bytes de cada campo, na parte superior.

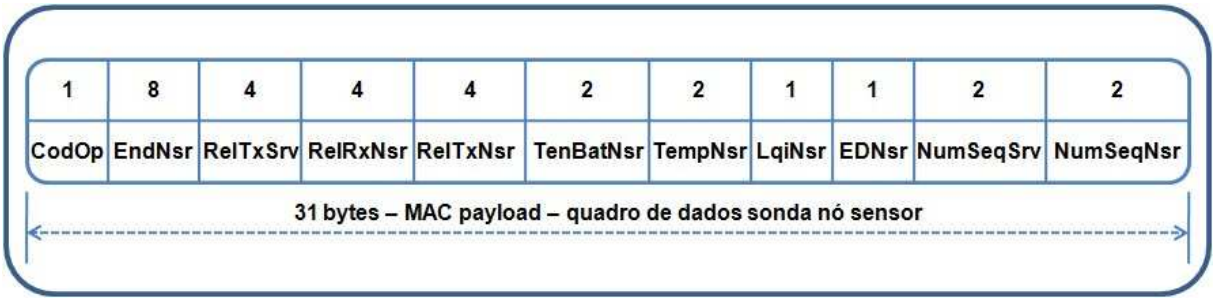


Figura 19 – Estrutura da carga útil do quadro de dados sonda do nó sensor

O quadro de dados sonda do nó sorvedouro é mais simples, e possibilita a radiodifusão de informações pela rede, ou envio do pacote a um nó específico. As informações adicionadas à carga útil da estrutura de dados da camada de acesso ao meio (MAC) são as seguintes:

- Código da operação (CodOp) – Idem ao primeiro campo do pacote sonda do nó sensor descrito anteriormente e ocupa 1 byte.

- Endereço do nó sensor de destino (EndNsr) – Representa o endereço do nó sensor de destino no padrão IEEE 802.15.4™-2006 LR-WPAN, composto por oito bytes. Sendo o valor 0xFF-FF-FF-FF-FF-FF-FF-FF reservado para as operações de radiodifusão, na qual uma mensagem é enviada para todos os nós sensores participantes da rede pessoal sem fio. Esse campo ocupa 8 bytes no pacote sonda.

- Relógio do nó sorvedouro no instante da transmissão do pacote sonda (RelTxSrv) – Representa o valor do relógio local do nó sorvedouro no instante da transmissão de um pacote sonda. Esse valor é utilizado para o cálculo da latência entre o envio e o recebimento do pacote sonda. Esse campo ocupa 4 bytes.

- Número de pacotes a serem enviados (NumPacEnv) – Representa a quantidade de pacotes a serem enviados pelo nó sensor ao nó sorvedouro. Esse campo é composto por dois bytes e possibilita a seleção entre 1 e 65.536 pacotes.

- Intervalo de envio dos pacotes (IntEnvPac) – Representa o intervalo entre o envio consecutivo de pacotes pelo nó sensor em milisegundos. Esse campo é composto por dois bytes e permite a seleção do intervalo entre 1 e 65.536 milisegundos.

- Número sequencial do pacote enviado (NumSeqSrv) – Identifica a sequência de pacotes transmitidos pelo nó sorvedouro, a cada pacote sonda enviado é inserido um campo com um número sequencial de 2 bytes. Esse campo é utilizado para avaliar perda na recepção de pacotes pelo nó sorvedouro.

A estrutura da carga útil do quadro de dados sonda do nó sorvedouro inserida no campo MAC payload, pode ser visualizada na figura 20, bem como a largura em bytes de cada campo na parte superior.

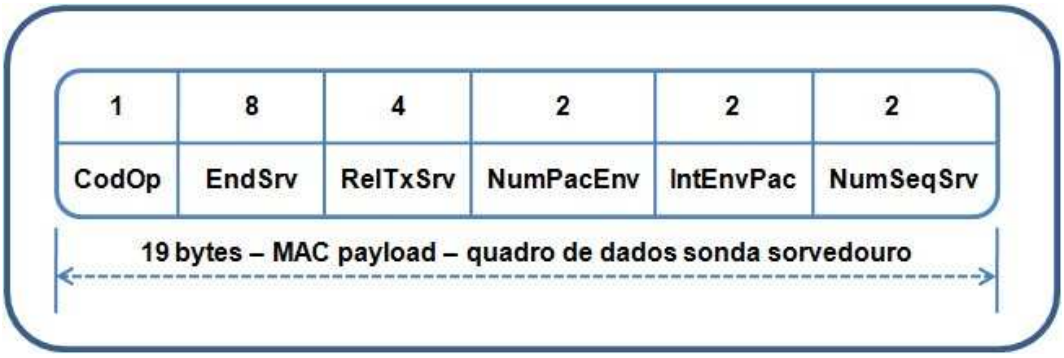


Figura 20 – Estrutura da carga útil do quadro de dados sonda do nó sorvedouro

As informações adicionadas ao pacote sonda, tanto do nó sorvedouro (Srv), quanto do nó sensor (Nsr), estão relacionadas no quadro 7 em ordem alfabética da sigla, contendo a descrição, o número do campo e o comprimento em bytes do campo no pacote sonda.

Sigla	Descrição	Campo		Bytes
		Srv	Nsr	
CodOp	Código da operação	1	1	1
EDNsr	Energia detectada pelo nó sensor no momento da recepção		9	1
EndNsr	Endereço do nó sensor		2	8
EndSrv	Endereço do sorvedouro	2		8
IntEnvPac	Intervalo de tempo entre o envio de pacotes pelo nó sensor	5		2
LqiNsr	Indicador do nível de qualidade da lig. no momento da recepção		8	1
NumPacEnv	Número de pacotes a ser enviado pelo nó sensor	4		2
NumSeqNsr	Número sequencial de pacotes enviados pelo nó sensor		11	2
NumSeqSrv	Número sequencial de pacotes enviados pelo sorvedouro	6	10	2
RelRxNsr	Relógio do nó sensor no instante da recepção do pacote		4	4
RelTxNsr	Relógio do nó sensor no instante da transmissão do pacote		5	4
RelTxSrv	Relógio do sorvedouro no instante da transmissão	3	3	4
TempNsr	Temperatura de operação do nó sensor		7	2
TenBatNsr	Tensão da bateria do nó sensor		6	2

Quadro 7 – Resumo das informações dos pacotes sonda

A alteração do programa fonte original do nó sensor para suportar a instrumentalização segue o fluxograma apresentado na figura 21 na qual são destacados os blocos funcionais do programa original e da instrumentalização de software. O programa original tem como funcionalidade o envio periódico de pacotes contendo informações de temperatura.

O fluxo original consiste de um loop infinito, cuja função é monitorar constantemente os eventos de interesse originalmente projetados. Na figura 21 esse fluxo original é representado em linhas pontilhadas e foi alterado com o intuito de introduzir as rotinas de instrumentalização de software.

O nó sensor fica monitorando em loop a ocorrência de eventos de interesse e o recebimento de pacotes. Ao receber um pacote, é feita a verificação se é um pacote sonda, identificado pelo campo código da operação (CodOp). Caso não seja um pacote sonda, o nó sensor continua em sua operação original, monitorando eventos e/ou aguardando novos pacotes. Caso o pacote recebido seja do tipo sonda, o nó sensor verifica qual é o tipo de operação a ser realizada, sendo que o código da operação (CodOp) 0x01 é utilizado para avaliação de aspectos temporais e 0x02

para avaliação da manutenção da qualidade da conexão. Se o código da operação não for nenhum desses valores, o pacote é descartado e o processo retorna ao estágio inicial.

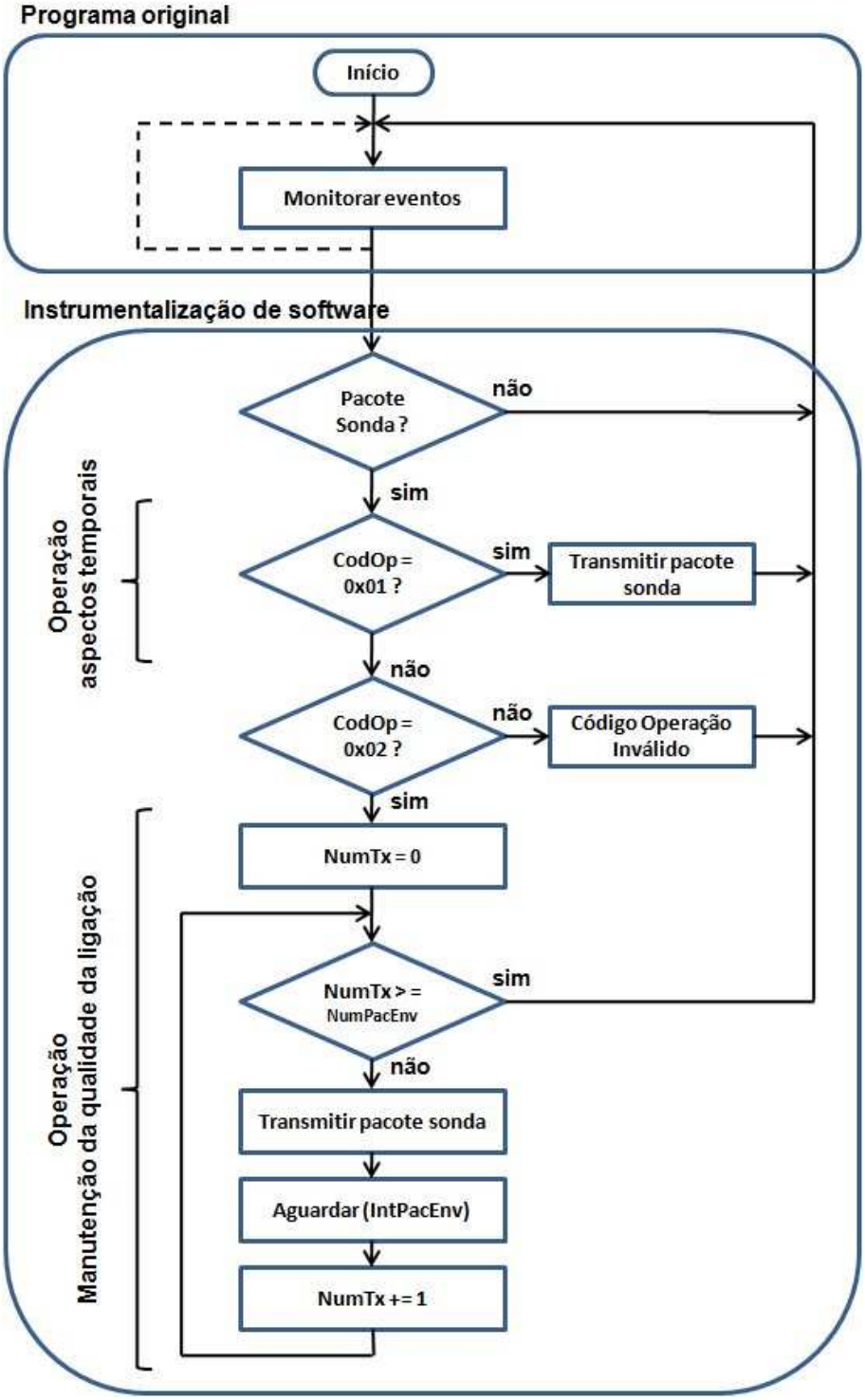


Figura 21 – Fluxograma de instrumentalização do nó sensor

Para a avaliação de aspectos temporais, código de operação 0x01, o nó sensor preenche todos os campos da estrutura do pacote sonda do nó sensor, conforme figura 19, e o envia de volta para o nó sorvedouro, retornando à etapa inicial.

Para a avaliação da manutenção da qualidade da conexão, código da operação 0x02, o nó sensor seleciona a quantidade de pacotes sonda a ser enviada ao nó sorvedouro (NumPacEnv) e também o intervalo de tempo a ser observado entre os envios (IntPacEnv). A partir desse momento o nó sensor passa a transmitir os pacotes sondas até alcançar a quantidade programada, observando o intervalo de tempo programado entre as transmissões. Ao final o processo retorna ao passo inicial.

A alteração do programa fonte do nó sorvedouro para suportar a instrumentalização segue o fluxograma da figura 22 na qual são destacados os blocos funcionais do programa original e da instrumentalização de software. O nó sorvedouro possui duas interfaces, uma para comunicação com o instrumento de software, descrito na próxima seção e outra para comunicação com os nós sensores.

A operação do nó sorvedouro é ficar monitorando o recebimento de pacotes da rede, bem como o recebimento de comandos enviados pelo instrumento de software. Caso seja recebido um pacote, o nó sorvedouro verifica se é um pacote evento ou um pacote sonda. Se for um pacote evento o nó sorvedouro continua sua operação original, identificada no fluxograma pela rotina “Tratar Pacote Evento”, retornando após essa operação ao estado de monitoração.

Caso o pacote recebido seja do tipo sonda, o nó sorvedouro realiza a rotina de formatação das informações a serem enviadas ao instrumento de software. Algumas informações de interesse só estão disponíveis no momento do recebimento do pacote sonda enviado pelos nós sensores e são relevantes para o levantamento do perfil da rede. Nesse sentido essas informações são reunidas em uma estrutura denominada de registro nó sorvedouro, na qual as informações são agrupadas em função da origem da informação, ou seja, provenientes do nó sensor ou do nó sorvedouro.

Caso seja recebido um comando enviado pelo instrumento de software, o nó sorvedouro realiza a formatação do pacote sonda e o transmite para a Rede de Sensores Sem Fio.

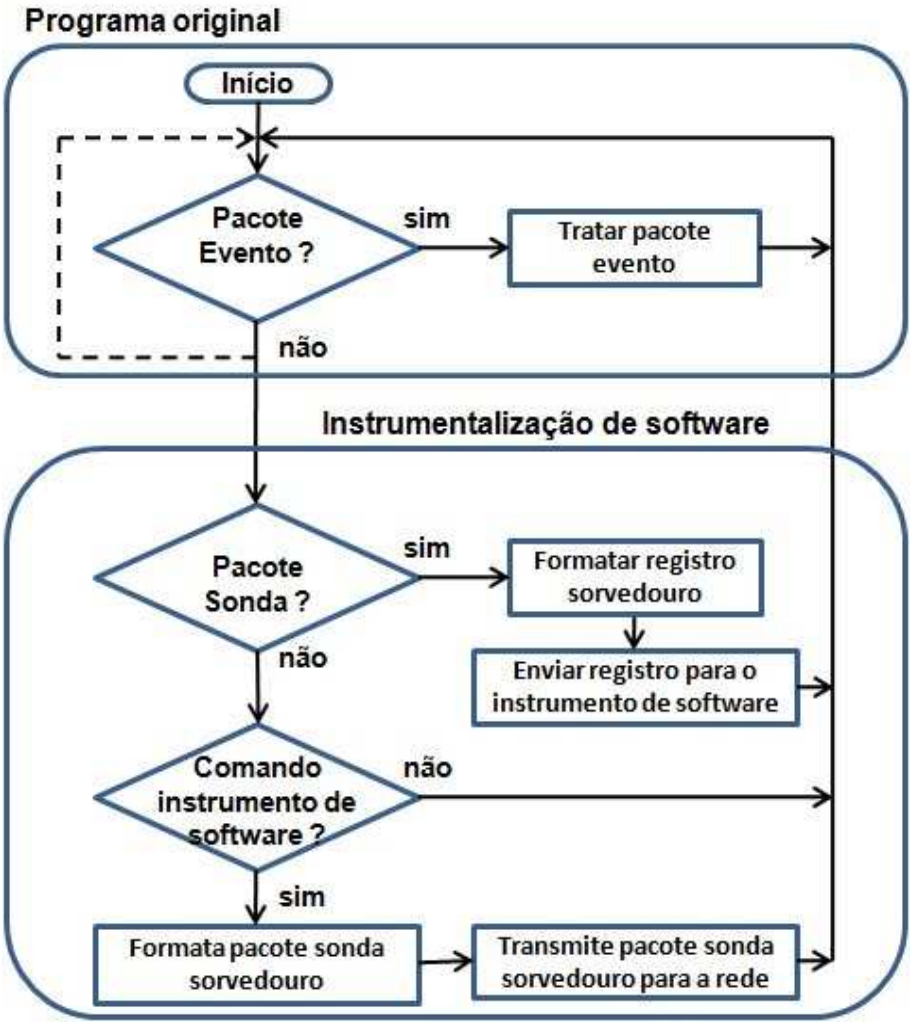


Figura 22 – Fluxograma de instrumentalização do nó sorvedouro

As informações obtidas do nó sorvedouro são:

- Endereço do nó sorvedouro (EndSrv);
- Valor do relógio do nó sorvedouro no instante da recepção (RelRxSrv);
- Valor do relógio do nó sorvedouro no instante da transmissão (RelTxSrv);
- Valor da tensão da bateria (TenBatSrv);
- Valor da temperatura (TempSrv);
- Valor do indicador da qualidade da conexão na recepção (LqiSrv);

- Valor da energia detectada na recepção (EDSrv) e
- Número sequencial do pacote enviado (NumSeqSrv).

As informações do nó sensor são obtidas do pacote sonda recebido são:

- Endereço do nó sensor (EndNsr);
- Valor do relógio do nó sensor no instante da transmissão (RelTxNsr);
- Valor do relógio do nó sensor no instante da recepção (RelRxNsr);
- Valor da tensão da bateria (TenBatNsr);
- Valor da temperatura (TempNsr);
- Valor do indicador da qualidade da conexão na recepção (LqiNsr);
- Valor da energia detectada na recepção (EDNsr); e
- Número sequencial do pacote enviado (NumSeqNsr).

A figura 23 ilustra a estrutura do registro, subdividida em informações do nó sorvedouro e do nó sensor.

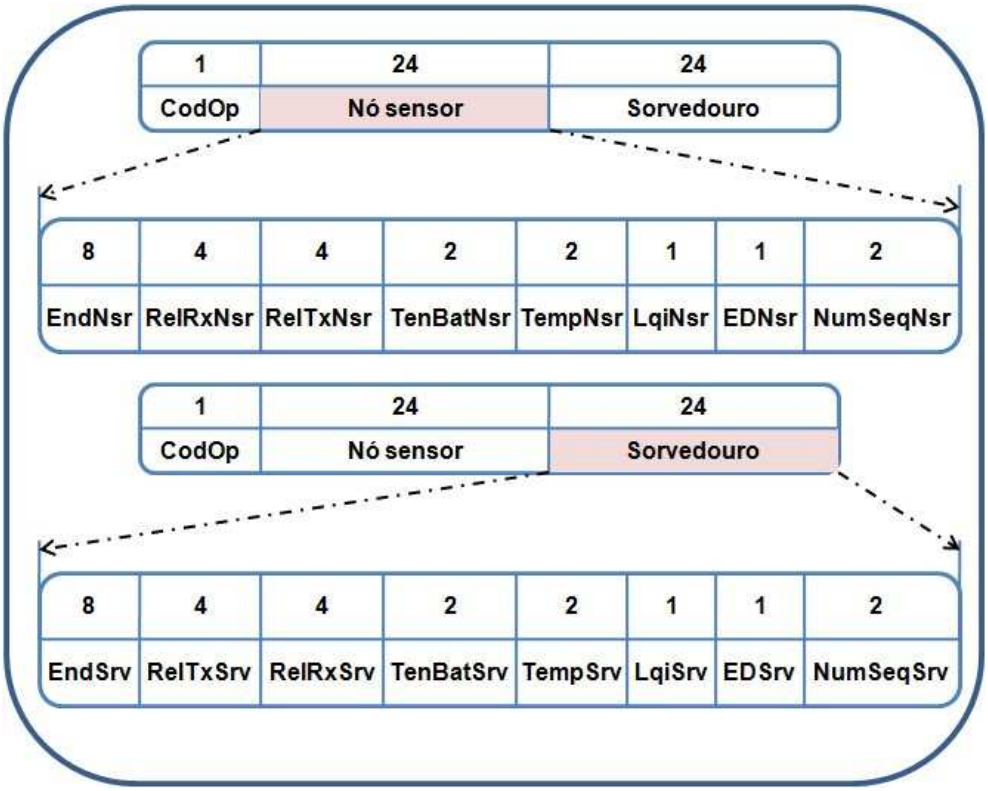


Figura 23 – Registro enviado pelo nó sorvedouro ao instrumento de software

Essas informações podem ser enviadas ao instrumento de software em três formatos diferentes:

- *American Standard Code for Information Interchange* (ASCII) que possibilita a visualização das informações em qualquer terminal ASCII independentemente do instrumento de software;

- Hexadecimal que reduz a quantidade de informações enviadas, sendo, portanto mais eficiente em termos de tempo de transmissão e,

- *Comma Separated Values* (CSV) que facilita a manipulação dos dados em aplicativos especializados em banco de dados.

5.2.2 Instrumento de Software

O instrumento de software é executado em um sistema computacional com recursos consideráveis, em termos de processador, de memória, de armazenamento e de energia, entre outros. Essas características suportam o processamento das mensagens sondas oriundas da rede sob análise. No presente trabalho é utilizada a ferramenta MatLab™, que possui as características necessárias para a criação do instrumento, possui suporte a comunicação serial, necessária para a interface com a Rede de Sensores Sem Fio, mais especificamente com o nó sorvedouro, coordenador da rede PAN. A ferramenta disponibiliza também recursos para a construção da interface gráfica, que é utilizada para a interação do operador com o instrumento e, também, possui recursos para manipulação de arquivos, necessários para a criação, abertura, leitura, escrita e fechamento de arquivos de dados.

O instrumento de software é composto por quatro módulos básicos, conforme ilustrado na figura 24 e descritos a seguir.

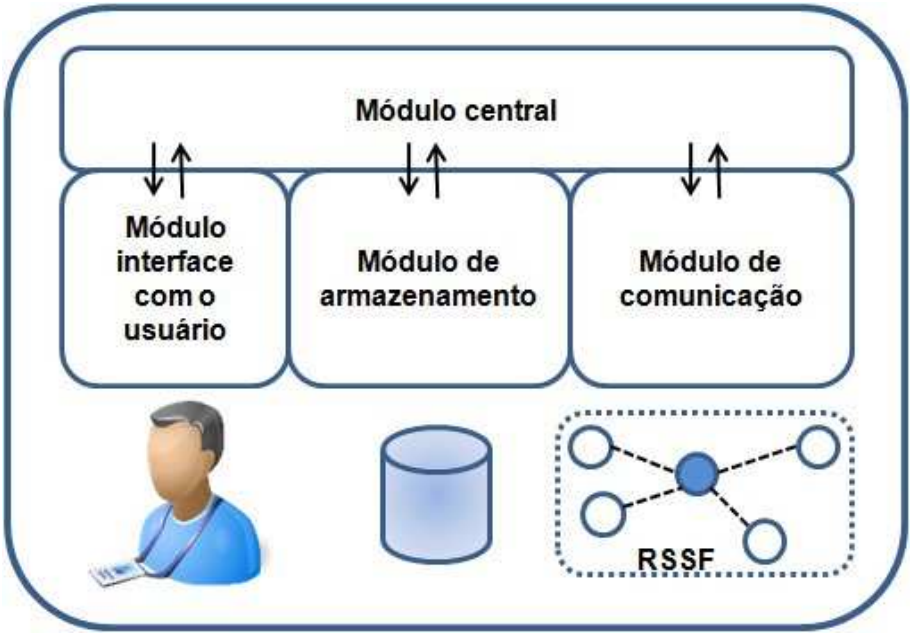


Figura 24 – Estrutura do instrumento de software

- Módulo de Interface com o Usuário – Responsável por receber comandos do usuário e mostrar o resultado das análises dos aspectos temporais e da manutenção da qualidade da conexão da rede. Possibilita ao usuário conhecer os perfis da Rede de Sensores Sem Fio.

- Módulo de Armazenamento – Responsável pelo gerenciamento da base de dados. Cada vez que o instrumento de software é executado, este módulo cria um arquivo com o nome codificado de forma que nunca sobrescreva um arquivo anterior, sendo que a codificação do nome é: RSSFAnoMêsDiaHoraMinuto.csv.

  Por exemplo, o arquivo RSSF201108011200.csv, trata-se de uma base de dados criada no dia 01/08/2011 às 12:00 horas. Após a criação, o módulo efetua as gravações e leituras de dados solicitadas pelo módulo central. Ao finalizar a execução do instrumento de software, o módulo de armazenamento efetua o fechamento do arquivo.

- Módulo de Comunicação – Responsável pelo envio de comandos e recebimento de respostas da Rede de Sensores Sem Fio. Esse módulo faz a abertura do canal de comunicação com a porta serial desejada e envia os

comandos solicitados pelo módulo central ao nó sorvedouro da rede. Outra atividade desenvolvida é o recebimento dos dados enviados pelo nó sorvedouro ao instrumento de software, repassando-os para o módulo central. Ao finalizar a execução do instrumento de software, o módulo de comunicação efetua o fechamento da porta de comunicação.

- Módulo Central – Responsável pelo processamento das informações com o objetivo de formatar os dados a serem armazenados e pelo cálculo dos valores a serem apresentados ao usuário via módulo de interface com o usuário.

  O Módulo Central trabalha de forma transparente no armazenamento do registro de dados recebidos do nó sorvedouro. Na prática este módulo acrescenta as informações de data e hora (*time stamp*) em cada registro recebido do nó sorvedouro, criando um novo registro a ser enviado ao módulo de armazenamento. A estrutura do registro é semelhante à da figura 23 acrescentando-se as informações de data no formato dd/mm/aaaa, de horário no formato hh:mm:ss:mmm e do símbolo vírgula (,) para delimitar os campos. A estrutura do registro armazenado pelo instrumento de software está ilustrada na figura 25.

Figura 25 – Estrutura do registro armazenado pelo instrumento de software

O instrumento de software disponibiliza funcionalidades para o usuário realizar o levantamento do perfil dos aspectos temporais e de manutenção da qualidade da conexão da rede.

Este instrumento envia comandos para o nó sorvedouro e recebe respostas e dados. Os comandos programados são para as avaliações dos aspectos temporais e de

manutenção da qualidade da conexão. Para ambos o instrumento de software envia ao nó sorvedouro o código da operação desejada (CodOp), o endereço do nó sensor de interesse ou um endereço especial para a seleção de todos os nós participantes da rede de sensores sem fio (EndNsr), a quantidade de pacotes a serem monitorados (NumPacEnv) e o intervalo de tempo a ser observado (IntEnvPac). De posse desses parâmetros o nó sorvedouro configura a rede para operar de acordo com a operação selecionada.

A operação para o levantamento do perfil dos aspectos temporais da rede consiste no envio do pacote sonda pelo nó sensor ao nó sorvedouro em resposta ao recebimento de uma solicitação do nó sorvedouro. O conceito utilizado é semelhante ao comando “ping[1]”, largamente utilizado em redes convencionais. O levantamento dos aspectos temporais da rede pode ser realizado para um nó sensor em particular ou para toda a rede.

A figura 26 ilustra o diagrama de relacionamento entre as entidades envolvidas no sistema para o levantamento dos aspectos temporais. O instrumento de software envia o comando para o nó sorvedouro, que por sua vez emite o pacote sonda para um determinado nó sensor (linha-seta cheia) ou envia o pacote sonda, no modo radiodifusão, para todos os nós sensores da rede (linha-seta pontilhada). O nó sensor decodifica o pacote sonda recebido e envia em resposta, imediatamente, o pacote sonda do nó sensor. O nó sorvedouro formata as informações recebidas da rede e repassa para o instrumento de software que por sua vez consolida os dados e realiza o armazenamento do registro para uma futura análise offline.

[1] O comando ping é amplamente utilizado para diagnosticar problemas em conexões *Transmission Control Protocol/Internet Protocol* (TCP/IP), através desse comando são enviados 4 pacotes de informações com o comprimento de 64 bytes ao servidor remoto solicitando que os devolva da mesma forma. Nessa operação é medido o tempo de ida-e-volta de um pacote ao destinatário, possibilitando dessa forma medir o tempo de resposta ao comando e se a estação remota está conectada (MICROSOFT, 2011).

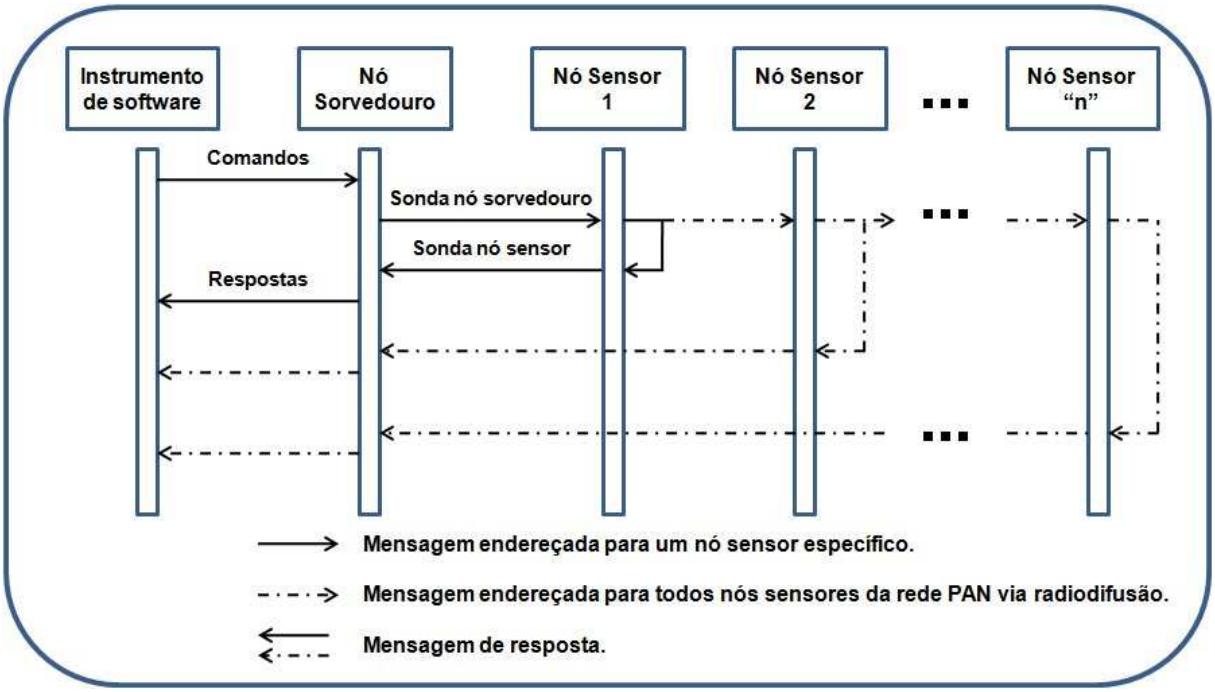


Figura 26 – Diagrama de relacionamento dos aspectos temporais da rede

A operação para o levantamento do perfil da manutenção da qualidade da conexão consiste no envio de uma quantidade programada de pacotes sondas pelos nós sensores ao nó sorvedouro. Existem duas formas de seleção do envio de pacotes, sendo que na primeira, o operador seleciona apenas um nó sensor da rede para levantar o perfil e na segunda, o operador seleciona a rede toda para o levantamento do perfil. Para atingir esse objetivo a solicitação é realizada por meio de radiodifusão do pacote sonda pelo nó sorvedouro pela rede. Assim todos os sensores passarão a enviar o pacote sonda ao nó sorvedouro.

A figura 27 ilustra o diagrama de relacionamento entre as entidades envolvidas no sistema. O instrumento de software envia o comando para o nó sorvedouro, que por sua vez emite o pacote sonda para um determinado nó sensor (linha cheia) ou envia o pacote sonda, no modo radiodifusão, para todos os nós sensores da rede (linha tracejada). O nó sensor decodifica o pacote sonda recebido e passa a enviar a quantidade de pacote sonda do nó sensor solicitada e em intervalos programados. O nó sorvedouro formata as informações recebidas da rede e repassa para o instrumento de software, que por sua vez consolida os dados e realiza o armazenamento do registro para uma futura análise offline.

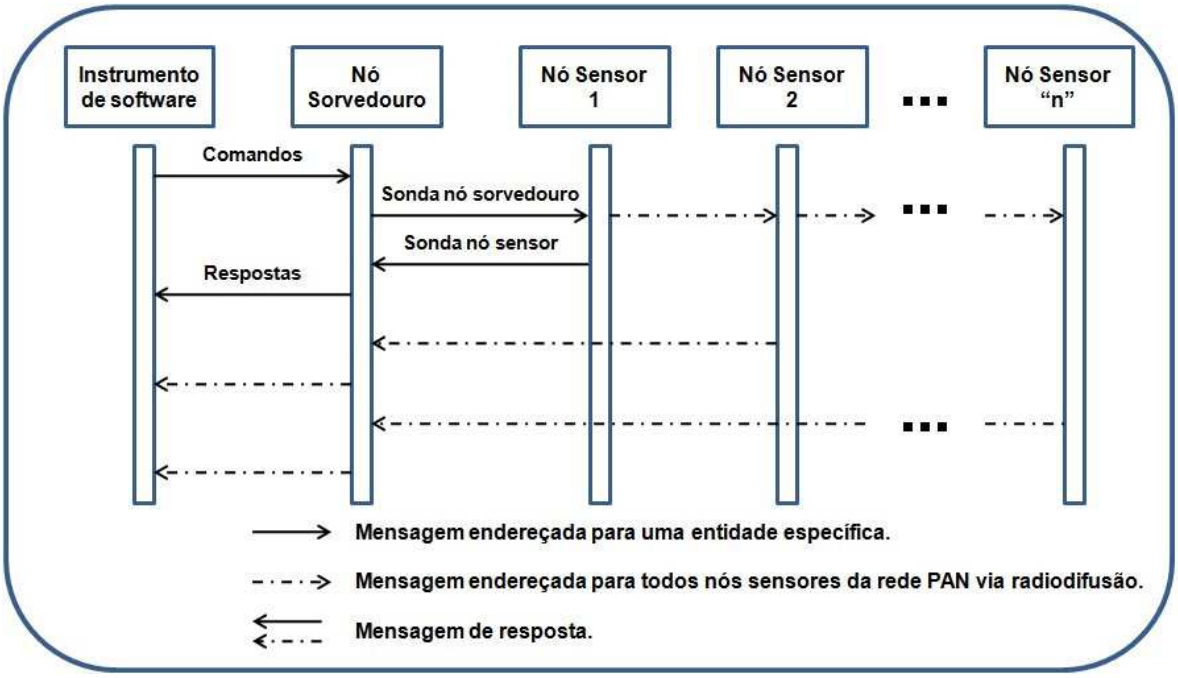


Figura 27 – Diagrama de relacionamento da manutenção da qualidade da conexão

5.3 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo foi apresentada a proposta de instrumentalização de software, em Redes de Sensores Sem Fio, suportada pelo padrão IEEE 802.15.4^{TM}-2006 LR-WPAN, em aplicações com restrições temporais. A instrumentalização envolve o desenvolvimento de um instrumento de software, e a instrumentalização dos programas fontes do nó sorvedouro e do nó sensor. A presente proposta visa proporcionar uma ferramenta que auxilie no levantamento do perfil da Rede de Sensores sem Fio, em termos dos aspectos temporais envolvidos e também da manutenção da qualidade da conexão.

No próximo capítulo é apresentado o projeto do ambiente de testes utilizado no levantamento de dados para a avaliação da instrumentalização de software proposta.

6 AMBIENTE DE TESTES

O objetivo do presente capítulo é descrever o ambiente de testes utilizado neste trabalho para a avaliação do levantamento do perfil de Redes de Sensores Sem Fio suportadas pelo padrão 802.15.4™-2006 LR-WPAN e utilizadas em aplicações com requisitos temporais, por meio da utilização do conceito de instrumentalização de software.

6.1 ARQUITETURA DO AMBIENTE DE TESTES

A Rede de Sensor Sem Fio avaliada possui uma topologia em estrela, composta de um coordenador de rede PAN, com a função adicional de nó sorvedouro e quatro nós sensores. A figura 28 ilustra a arquitetura do ambiente de testes utilizado no presente trabalho. Essa topologia foi adotada em função de sua simplicidade, pois possui uma conexão direta entre os nós sensores e o nó sorvedouro, simplificando as atividades de desenvolvimento do software da Rede de Sensor sem Fio, em função de não haver a necessidade de se criar algoritmos de roteamento o que foge ao escopo do presente trabalho.

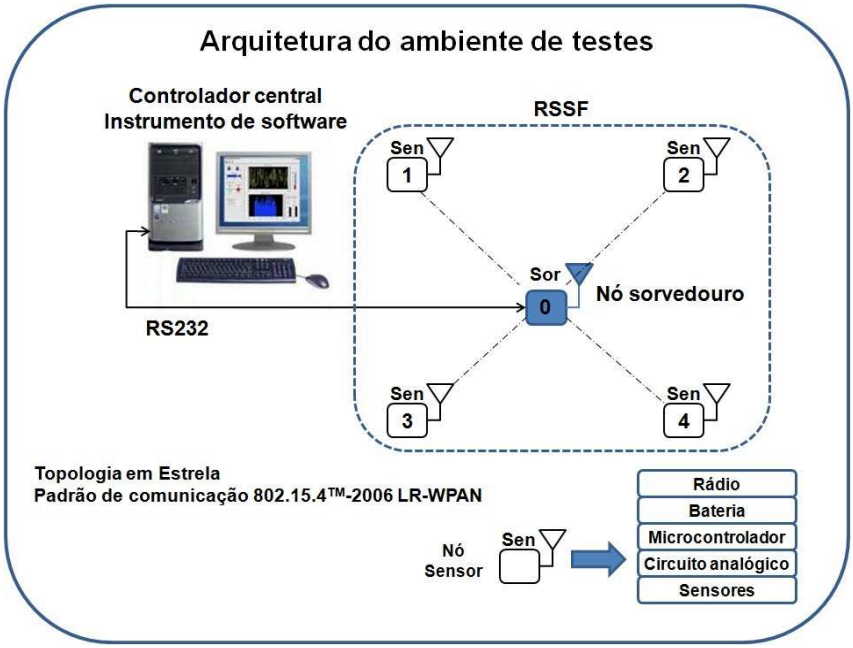


Figura 28 – Arquitetura do ambiente de testes

O ambiente de testes é constituído pelos seguintes elementos: instrumento de software e a Rede de Sensores sem Fio, constituída por um nó sorvedouro e quatro nós sensores.

6.2 INSTRUMENTO DE SOFTWARE

O instrumento de software é um programa que é executado em um computador com o objetivo de complementar a instrumentalização da Rede de Sensores sem Fio. O instrumento de software desempenha as funções de aquisitar, tratar, armazenar e avaliar dados enviados pelo nó sorvedouro, computando itens relacionados aos aspectos temporais e também aos aspectos da manutenção da qualidade da conexão. Para construção do instrumento de software que realize as funções citadas, foi utilizada a ferramenta MatLab™, da empresa MathWorks. Essa ferramenta é utilizada para a solução e visualização de problemas numéricos, possuindo um ambiente interativo cujos recursos facilitam o desenvolvimento de aplicações típicas de engenharia. O MatLab™, possibilita sua integração com outras linguagens de alto nível, como C, C++, JAVA e Fortran. O MatLab™, possui ainda uma ampla gama de funcionalidades voltada para a solução de problemas específicos, conhecida como “toolboxes”. Outro atrativo desse ambiente é a facilidade de desenvolvimento de interface gráfica para interação com o usuário.

A figura 29 apresenta a arquitetura de software utilizada para a instrumentalização de uma Rede de Sensores Sem Fio. Nessa figura o bloco Instrumento de Software é o responsável pela interface com o usuário. Os blocos Nó Sorvedouro e Nó Sensor contemplam o programa original acrescentado das rotinas de instrumentalização.

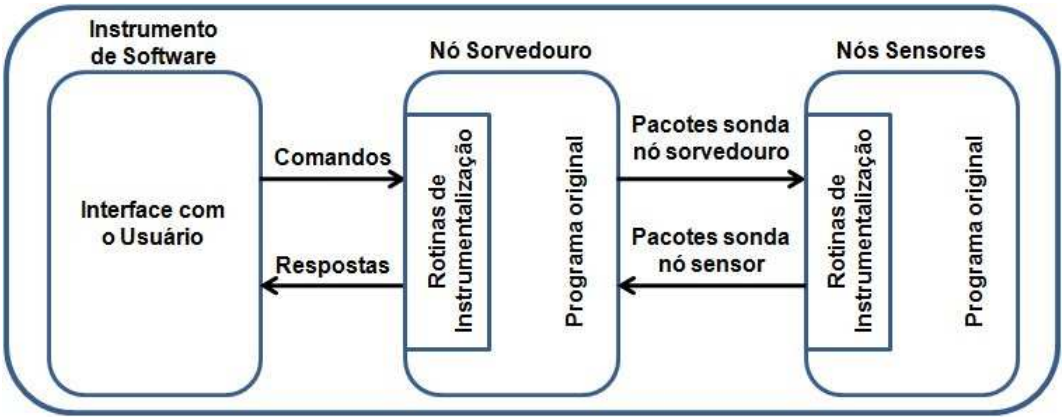


Figura 29 - Arquitetura de software da instrumentalização de Redes de Sensores Sem Fio

6.2.1 Hardware do Instrumento de Software

O hardware do controlador central é composto por um microcomputador do tipo notebook, equipado com um processador Pentium M e frequência de operação de 1500 MHz, com 512 MBytes de memória RAM, disco rígido de 40 GBytes, interface para redes *Wireless Fidelity* (Wi-Fi) padrão 802.11, leitor e gravador de *Compact Disk – Read Only Memory* (CD-ROM) e adaptador para interface serial RS232. Este hardware foi escolhido por estar disponível, e por permitir a mobilidade do ambiente de teste para os diversos locais, o que facilita a instalação do ambiente nas áreas de interesse. O controlador central possui os seguintes softwares instalados: sistema operacional Windows XP Professional, o instrumento de software e outros softwares de suporte ao ambiente de testes. O instrumento de software é utilizado na aquisição de dados, processamento desses dados, apresentação de forma conveniente dos mesmos ao usuário e também no armazenamento de dados.

6.2.2 IHM - Interface Homem Máquina

A interface com o usuário utiliza os recursos *Graphical User Interface* (GUI) do MatLab™. O papel básico desse módulo é criar um ambiente interativo no qual o usuário possa configurar os parâmetros necessários do instrumento de software e

também permitir a visualização das medições realizadas na Rede de Sensores sem Fio sob avaliação, possibilitando o conhecimento do perfil da rede. A IHM permite ao usuário configurar qual o perfil a ser levantado, ou seja, o perfil dos aspectos temporais ou o perfil da manutenção da qualidade da conexão da rede. Outro parâmetro necessário é a configuração do endereço do nó sensor de interesse, podendo ser configurado um nó sensor em particular ou todos os elementos constituintes da rede, por meio de um endereço especial denominado de radiodifusão.

Outros parâmetros configurados são o intervalo de envio dos pacotes e a quantidade de pacotes a serem enviada pelo sorvedouro ou pelo nó sensor. A figura 30 ilustra a IHM utilizada para a configuração dos parâmetros do instrumento de software.

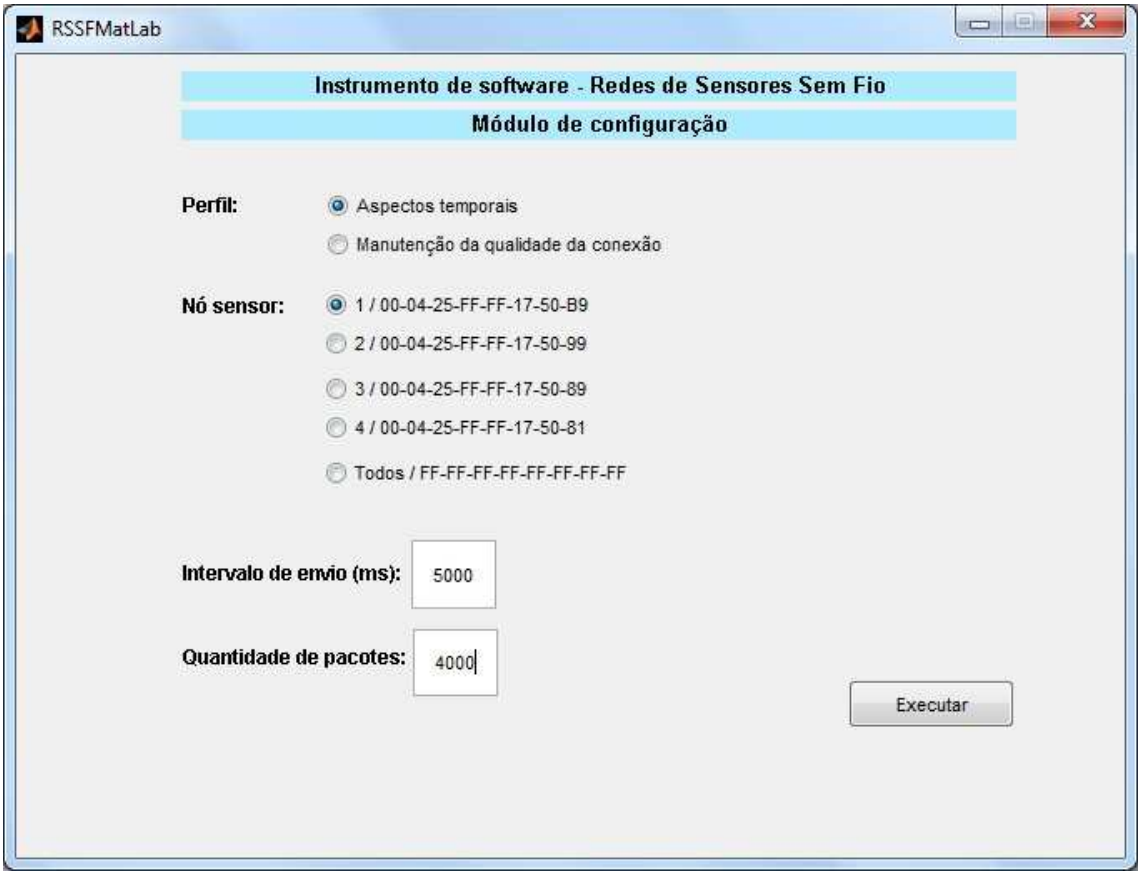


Figura 30 – IHM – Módulo interface com o usuário - configuração

Para visualização do perfil dos aspectos temporais, é utilizada a interface ilustrada na figura 31. Nessa interface é possível verificar os valores do tempo de ida-e-volta (RTT), classificados como atual, que é o último valor medido pelo instrumento de software, médio, mínimo e máximo.

Figura 31 – Módulo interface com o usuário – aspectos temporais

A interface visualizada na figura 32 permite ao usuário o conhecimento dos aspectos da manutenção da qualidade da conexão. Nessa interface é possível conhecer o comportamento da rede em termos de energia detectada (ED), indicador da qualidade da conexão (LQI), tensão da bateria (TB) e temperatura de operação (TO) de forma semelhante a figura anterior. A taxa de pacotes recebidos (PRR) indica o porcentual de pacotes efetivamente recebidos pelo nó sorvedouro.

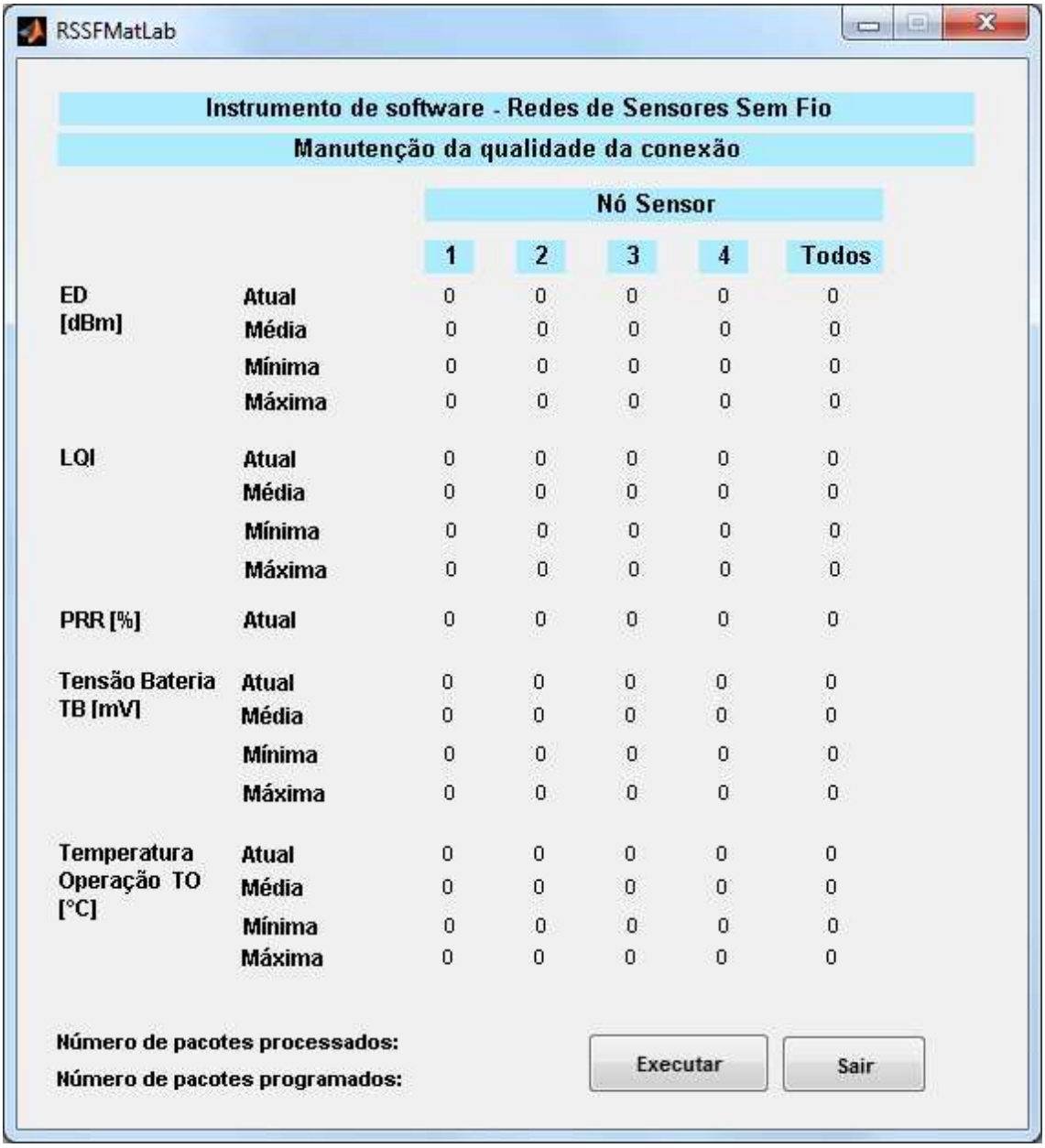


Figura 32 – Módulo interface com o usuário – manutenção da qualidade da conexão

6.2.3 Armazenamento de dados

O módulo de armazenamento é responsável pela gravação do registro gerado pelo instrumento de software. Esse módulo utiliza as funções de manipulação de arquivos do MatLab™ para criar o arquivo desejado, gravar o registro formatado com as informações recebidas do sorvedouro e ao final fechar o arquivo. A figura 33 mostra o pseudocódigo deste módulo.

Pseudocódigo módulo armazenamento

```
/* Abertura do arquivo RSSF201108011200.csv */
   fid = fopen('RSSF201108011200.csv', 'r+');

   /* loop */
   reg = strcat(serial_dataread, ';', datestr(now));

   fwrite(fid, reg);
   /* end loop */

   /* Fechamento do arquivo
   fclose('RSSF201108011200.csv');
```

Figura 33 – Pseudocódigo do módulo de armazenamento de dados

6.2.4 Comunicação serial com o nó sorvedouro

O módulo de comunicação utiliza a porta serial, no padrão RS232 identificada por “com2” no instrumento de software. Os parâmetros de configuração utilizados podem ser visualizados no quadro 8.

Parâmetro	Valor
Identificação da porta serial	COM2
Taxa de comunicação (bps)	9600
Bits de dados	8
Paridade	Nenhum
Bits de parada	1
Controle de fluxo	Nenhum

Quadro 8 – Características configuradas da porta de comunicação serial

O pseudocódigo do módulo de comunicação pode ser verificado na figura 34, no qual é definido um manipulador para a porta de comunicação, definido como “s”. Os valores programados para os parâmetros listados no quadro 8 são configurados por meio do comando “set” e logo em seguida é realizada a abertura da porta por meio do comando “fopen”. A leitura dos dados presentes na porta serial é realizada pelo comando “fscanf” que retorna os dados lidos em uma cadeia de caracteres (serial_dataread) e também a quantidade de caracteres lidos (count). O software fica

em loop lendo os dados da interface serial até o término do processo, quando então é feito o fechamento da porta e liberação dos recursos.

Pseudocódigo módulo comunicação

```
s   = serial('com2');
set(s, 'BaudRate', 9600, 'Timeout', 30, 'Terminator', '<');
fopen(s);
 get(s);

/* loop */
[serial_dataread, count] = fscanf(s);
/* end loop */

fclose(s);
delete(s);
clear s;
```

Figura 34 – Pseudocódigo do módulo de comunicação

6.3 REDE DE SENSORES SEM FIO INSTRUMENTALIZADA

A Rede de Sensores sem Fio utilizada no ambiente de testes é composta pelos nós sensores e pelo nó sorvedouro. Esta seção descreve as características de hardware desses dois dispositivos, o ambiente de desenvolvimento de software utilizado e a instrumentalização de software realizada na rede.

6.3.1 Hardware do nó sensor

O hardware do nó sensor é baseado no circuito integrado ATmega128RFA1 da empresa ATMEL, lançado em 2009. Esse componente é resultado da integração do microcontrolador ATmega1281 e do transceptor de rádio AT86RF231 e disponibiliza os padrões 802.15.4™ e ZigBee™ em um único circuito integrado (ATMEL, 2009). O microcontrolador possui alto desempenho e baixo consumo de energia, possuindo uma arquitetura de 8 bits e um conjunto reduzido de instruções, denominadas de

Reduced Instruction Set Computing (RISC), que disponibiliza 135 instruções, que na sua grande maioria são executadas em apenas um ciclo de relógio. Como o relógio do nó sensor opera a 16 MHz, as instruções levam 62,5 ns cada uma para ser processada.

Há três tipos de memória. A primeira é a memória destinada à gravação do programa do usuário e utiliza uma tecnologia flash, com capacidade de 128 KBytes. A segunda é a memória destinada à gravação de parâmetros fixos, como, por exemplo, o endereço do nó sensor, e utiliza a tecnologia *Electrically-Erasable Programmable Read-Only Memory* (EEPROM), com uma capacidade de 4 KBytes. Ambas são não voláteis, ou seja, elas mantêm os dados armazenados mesmo na falta de energia elétrica. A terceira é a memória de trabalho e utiliza a tecnologia *Static Random Access Memory* (SRAM) com uma capacidade de 16 KBytes, sendo que esta última é uma memória do tipo volátil. As principais características desse microcontrolador são as seguintes:

- Interface para testes padrão IEEE 1149.1, denominada de *Join Test Action Group* (JTAG);

- Canais múltiplos para temporizadores, contadores e moduladores de largura de pulso – *Pulse Width Modulation* (PWM);

- Contador de tempo real com oscilador independente;

- Conversor analógico digital com resolução de 10 bits e taxa de amostragem de 330 Ksamples/s;

- Duas interfaces seriais programáveis do tipo *Universal Synchronous and Asynchronous Receiver and Transmitter* (USART);

- Temporizador do tipo *Watchdog* com oscilador independente;

O transceptor de rádio é especificado para operar na faixa ISM com uma frequência de 2.4 GHz, suporta várias taxas de transmissão e recepção de dados, sendo utilizada a taxa de 250 Kbps no presente trabalho, possui uma sensibilidade de

recepção de -100 dBm e uma potência de transmissão de até +3,5 dBm, fornecendo suporte de hardware para o protocolo da camada de acesso ao meio (MAC) e outras funções dedicadas. Informações completas sobre o componente podem ser encontradas no manual de referência do fabricante (ATMEL, 2009). O departamento de suporte do fabricante recomenda a utilização de uma placa controladora de rádio desenvolvida pela empresa alemã “Dresden-Elektronik”, a qual contém, além do circuito integrado especificado anteriormente, três diodos emissores de luz – *Light Emitting Diode* (LED) e uma micro-chave do tipo botão. Esses elementos podem ser programados para as mais diversas finalidades e servem como interface com o usuário. A Atmel recomenda a utilização desse hardware e disponibiliza vários programas de demonstração que são suportados por essa plataforma. A figura 35 apresenta a placa eletrônica com a identificação dos principais componentes e também suas dimensões.

A placa disponibiliza ainda dois conectores de expansão que servem para acessar as funcionalidades do componente possibilitando a customização do circuito para fins específicos. O esquema elétrico da placa RCB128RFA1-V6.3.1 pode ser encontrado no anexo-A. O software do nó sensor é dedicado e desenvolvido em linguagem C, utilizando o ambiente integrado de desenvolvimento denominado de AVR Studio. Esse ambiente é prático e permite a edição do programa, a compilação, o carregamento do programa na placa controladora de rádio por meio da interface JTAG e também, a depuração do programa de forma integrada.

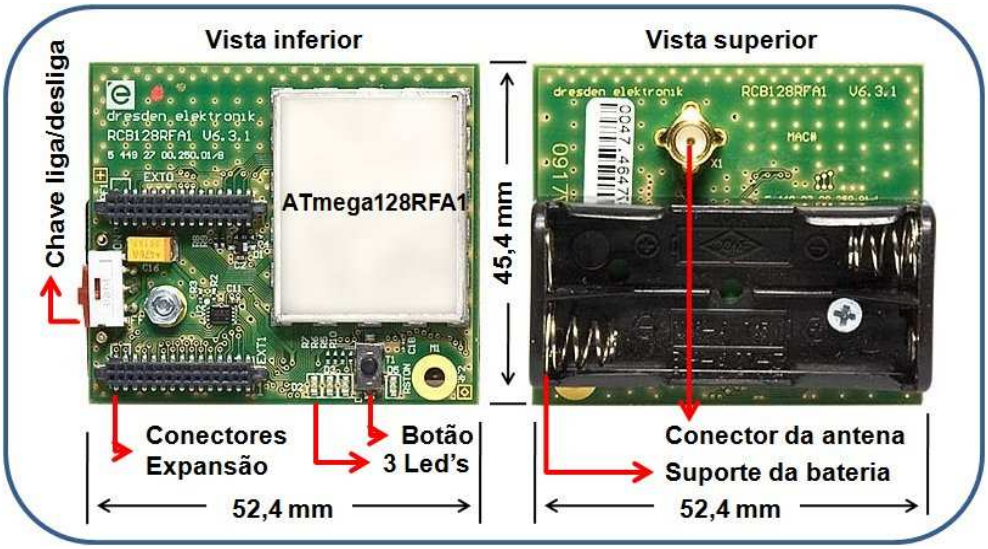


Figura 35 – Nó sensor baseado no circuito integrado ATmega128RFA1

6.3.2 Hardware do nó sorvedouro

O hardware do nó sorvedouro é idêntico ao hardware do nó sensor, porém para possibilitar a comunicação com o controlador central, por meio da interface serial RS232 é necessária a utilização de uma placa adicional projetada pela empresa alemã “Dresden-Elektronik”. Essa placa é conectada ao hardware do nó sensor por meio dos dois conectores de expansão. Essa placa disponibiliza uma interface com o ambiente integrado de desenvolvimento por meio do padrão JTAG e também uma interface com a comunicação serial RS232, a qual é utilizada para interligação do nó sorvedouro com instrumento de software.

Esse hardware também permite o acesso a várias funcionalidades da placa controladora de rádio por meio de dois conectores do tipo barra de pinos, identificados como “interface funcional”. A figura 36 ilustra a placa de interligação identificando as interfaces disponíveis e destacando também suas dimensões. O esquema elétrico da placa de interligação – *Radio Control Board Breakout Board* (RCBBB) pode ser encontrado no anexo-B. Adicionalmente, é necessária a utilização de um cabo de comunicação serial para a interconexão do nó sorvedouro com o instrumento de software. A união da placa de interligação e o hardware do nó sensor é denominada de nó sorvedouro. Em relação ao software do nó sorvedouro, valem as mesmas observações feitas para o nó sensor, porém o software é diferenciado, pois as funções do nó sorvedouro são diferentes das funções dos nós sensores.

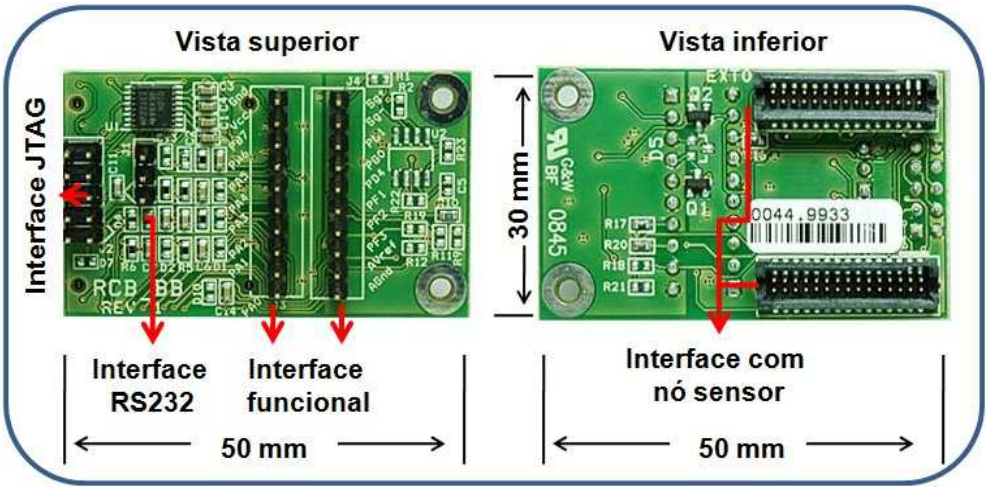


Figura 36 – Placa interligação nó sensor

6.3.3 Ambiente de desenvolvimento de software

A programação e a depuração dos programas dos nós sensores e do nó sorvedouro são realizadas por meio da utilização de um ambiente integrado de desenvolvimento composto por um dispositivo depurador e programador, denominado de “AVR JTAGICE mkII”, o qual disponibiliza uma interface JTAG para interligação com os nós sensores e também uma interface USB para interligação com o computador utilizado como estação de desenvolvimento. A figura 37 destaca o dispositivo JTAGICE mkII interligado com o nó sensor por meio da placa de interligação RCBBB e utilizando a interface JTAG.

Faz parte desse ambiente de desenvolvimento o software “AVR Studio” que é o ambiente integrado para desenvolvimento de aplicativos para microprocessadores da família AVR da empresa Atmel. Esse software utiliza um conjunto de ferramentas de código aberto (*open source*), denominada de WINAVR, o qual foi desenvolvido especificamente para a família de microprocessadores RISC da série AVR, disponibilizando bibliotecas de funções e compiladores para as linguagens C e C++, entre outras facilidades (O’FLYNN, 2011). A integração dessas ferramentas incluindo o dispositivo depurador e programador, o ambiente de desenvolvimento AVR Studio e o WINAVR proporciona a criação de um ambiente de desenvolvimento atrativo devido ao custo, das funcionalidades disponíveis e da curva de aprendizado da utilização do ambiente ser relativamente suave.

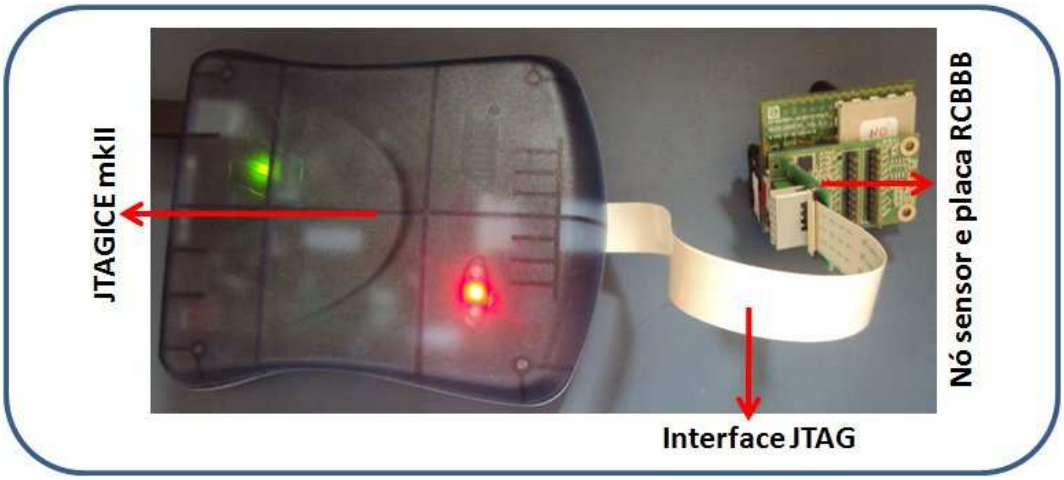


Figura 37 – Ambiente de desenvolvimento

6.3.4 Instrumentalização dos nós sensores e do nó sorvedouro

O desenvolvimento da instrumentalização de software da Rede de Sensores sem Fio foi suportado pela aplicação denominada de *MAC Example – Basic Sensor Network*, disponível no pacote de software “The MAC Software Package” que é utilizada como referência para construção de Redes de Sensores sem Fio. O pacote de software serve especificamente para a demonstração do uso dos componentes de comunicação sem fio da empresa Atmel que são aderentes ao padrão IEEE 802.15.4^{TM}-2006 LR-WPAN e geralmente é utilizado como modelo para o desenvolvimento de aplicações utilizando a plataforma Atmel (ATMEL, 2010).

A aplicação original, doravante denominada de rede não instrumentalizada, foi desenvolvida por meio da utilização de funções de acesso à camada de controle de acesso ao meio (MAC), denominadas de API. A rede é constituída de dispositivos que são nós sensores, nós que podem desempenhar funções de roteadores e nó sorvedouro. Utilizando diferentes tipos de nós é possível construir uma rede com topologia em árvore ou estrela. O roteamento da rede é estático, sendo que a topologia da rede e o roteamento são fixados no momento da criação da rede.

A rede possui uma identificação pré-definida, como sendo (0xBEEF) e utiliza o canal de comunicação 20. O funcionamento dessa rede se resume na coleta, pelos nós sensores, da tensão de alimentação de cada nó e também da temperatura de operação e sua transmissão para o nó sorvedouro, diretamente ou via nó roteador, em intervalos de 10 segundos. Os dados recebidos pelo nó sensor são enviados para a porta serial USART/USB. O software foi desenvolvido utilizando um único programa fonte e a decisão se o nó irá funcionar como sensor, roteador ou sorvedouro é realizada no momento de criação da rede. Se o botão ilustrado na figura 35 estiver acionado durante a energização do nó, ele pode desempenhar os papéis de sorvedouro ou roteador. O tipo selecionado vai depender se a rede já possuir um nó sorvedouro. Caso negativo o nó é configurado como sorvedouro e caso já exista um nó sorvedouro ele se configura como nó roteador. Caso o botão não esteja acionado durante a energização do nó ele assume as configurações de um nó sensor (ATMEL, 2010).

Utilizando a aplicação *Basic Sensor Network* como plataforma, foram realizadas alterações para a utilização do conceito de instrumentalização de software. O programa fonte foi originalmente escrito em linguagem C e foram desenvolvidos dois programas fontes, um para instrumentalizar o nó sorvedouro e outro para instrumentalizar os nós sensores.

O primeiro programa, denominado de “sorvedouro.c” com o objetivo de instrumentalizar o nó sorvedouro para que operasse funcionalmente de acordo com o fluxograma apresentado na figura 22. O pseudocódigo do programa “sorvedouro.c” pode ser visualizado na figura 38, no qual foi feita a instrumentalização necessária para a operação da rede utilizando esse conceito. O anexo-C apresenta o código do programa sorvedouro.c no qual as linhas 2, 3, 4, 5, 13, 14, 18 e 19 do pseudocódigo apresentado estão destacadas com o objetivo de facilitar a localização das rotinas de instrumentalização inseridas no programa. Primeiro foi desenvolvido um módulo de inicialização para configurar as características da rede, como topologia, identificação da rede, canal utilizado e potência de transmissão, entre outros. Depois foi necessário um módulo que configura o componente para desempenhar as funções de nó sorvedouro, com rotinas de leitura e escrita da porta seria USART, rotina de criação da rede, como descoberta de novos integrantes e, finalmente, foi desenvolvida uma mecânica que permita a agregação de novos nós sensores na rede por meio do acionamento do botão disponível no componente.

O segundo programa, denominado de “sensor.c”, com o objetivo de instrumentalizar o nó sensor para que operasse funcionalmente de acordo com o fluxograma apresentado na figura 21. O pseudocódigo do programa “sensor.c” pode ser visualizado na figura 39. O anexo-D apresenta o código do programa sensor.c no qual as linhas 2, 5, 6, 8, 14, 15, 17 e 18 do pseudocódigo apresentado estão destacadas com o objetivo de facilitar a localização das rotinas de instrumentalização inseridas no programa.

```
1.  .Inicializar as características da rede.
2.  .Configurar o hardware para operar como nó sorvedouro.
3.  .Se o botão estiver pressionado.
4.      .Enviar mensagem para descoberta de novos nós sensores.
5.  .Se o Comando do instrumento de software foi recebido.
6.      .Tratar comando do instrumento de software (linha 13).
7.  .Se o Pacote de Dados foi recebido.
8.      .Se o pacote de dados recebido é do tipo EVENTO.
9.          .Tratar pacote de evento (linha 16).
10.     .Se o pacote de dados recebido é do tipo SONDA.
11.         .Tratar pacote de sonda (linha 18).
12. .Voltar para a (linha 3).
13. .Formatar pacote sonda sorvedouro.
14. .Transmitir pacote sonda sorvedouro para a rede.
15. .Voltar para a (linha 3).
16. .Tratar pacote EVENTO na forma original da rede.
17. .Voltar para a (linha 3).
18. .Formatar registro sorvedouro com dados de interesse.
19. .Enviar registro sorvedouro para o instrumento de software.
20. .Voltar para a (linha 3).
```

Figura 38 – Pseudocódigo do programa "sorvedouro.c"

```
1.  .Inicializar as características da rede..
2.  .Configurar o hardware para operar como nó sensor.
3.  .Se Evento for detectado.
4.      .Tratar evento (linha 11).
5.  .Se o pacote sonda sorvedouro for recebido.
6.      .Se o pacote sonda for levantamento perfil aspectos temporais da
    rede.
7.          .Tratar aspectos temporais (linha 14).
8.      .Se o pacote sonda for levantamento perfil manutenção da
    qualidade da conexão da rede.
9.          .Tratar manutenção da qualidade da conexão (linha 17).
10. .Voltar para a (linha 3).
11. .Formatar pacote evento nó sensor.
12. .Transmitir pacote evento nó sensor para o sorvedouro.
13. .Voltar para a (linha 3).
14. . Formatar pacote sonda nó sensor.
15. .Transmitir pacote sonda nó sensor para o sorvedouro.
16. .Voltar para a (linha 3).
17. . Formatar pacote sonda nó sensor.
18. . Transmitir pacote sonda nó sensor para o sorvedouro.
19. .Se numero de pacotes sonda nó sensor enviados for igual ao solicitado
20.     .Voltar para a (linha 3).
21. .Aguardar intervalo de tempo programado.
22. .Voltar para a (linha 17).
```

Figura 39 – Pseudocódigo do programa “sensor.c”

Vale ressaltar que a estrutura do software original, da rede não instrumentalizada, sofreu mudanças significativas em virtude da substancial alteração de forma de funcionamento, passando de uma rede programada meramente para o envio periódico de pacotes de dados para uma rede configurável para avaliação do perfil dos aspectos temporais envolvidos e também do perfil da manutenção da qualidade da conexão da rede.

A figura 40 proporciona uma visão sistêmica dos elementos utilizados para a construção do ambiente de testes, sendo separado por hardwares, softwares e usuário. A interação com o usuário é realizada através do módulo interface gráfica (GUI) do software MatLab™. O MatLab™ é utilizado no desenvolvimento do instrumento de software, com suporte da linguagem C e que utiliza como hardware um computador.

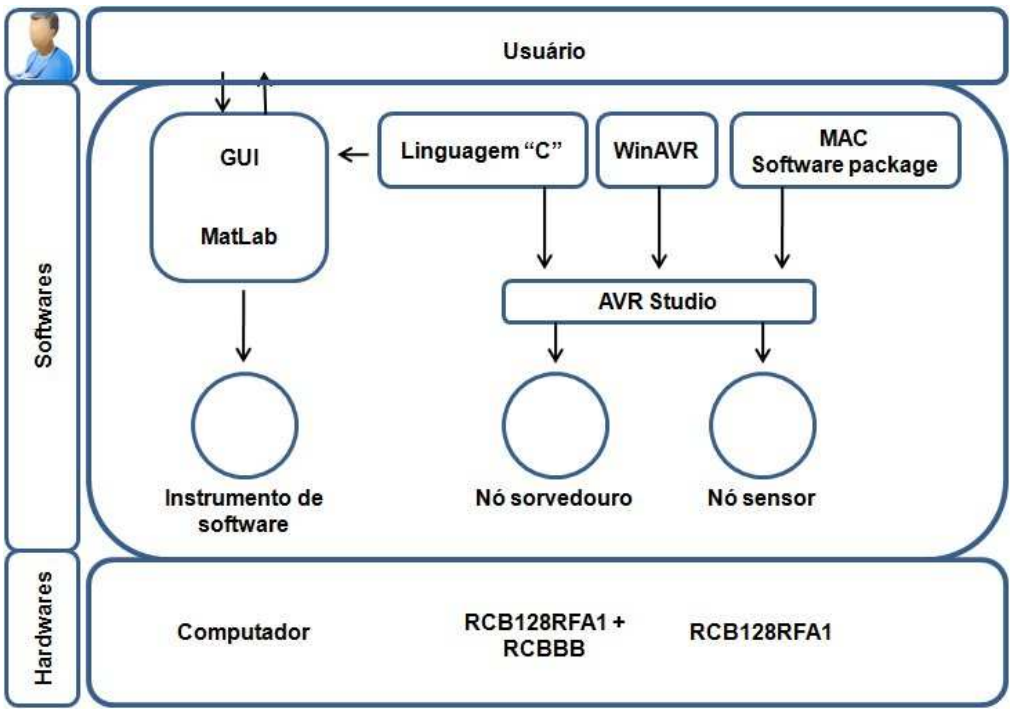


Figura 40 – Visão sistêmica dos componentes de hardware e software utilizados na construção do ambiente de testes

Para o desenvolvimento dos softwares do nó sorvedouro e do nó sensor são utilizados o ambiente de desenvolvimento integrado – *Integrated Development Environment* (IDE) identificado como AVR Studio, que utiliza a plataforma de

compilação WinAVR e o ambiente de suporte ao padrão IEEE 802.15.4™-2006 LR-WPAN, identificado como MAC software package.

O software do nó sorvedouro é embarcado no hardware RCB128RFA1 em conjunto com a placa de interligação RCBBB. O software do nó sensor é embarcado no hardware RCB128RFA1.

6.4 CARACTERÍSTICAS DA REDE DE SENSORES SEM FIO UTILIZADA NO AMBIENTE DE TESTES

As características da camada física (PHY) selecionadas foram o canal de operação, mais especificamente o canal 20, que tem 2450 MHz como frequência central, e uma banda passante de 2 MHz, conforme pode ser verificado na figura 41, na qual o canal 20 está destacado. O padrão utiliza o espalhamento espectral por sequência direta (DSSS) e o método de modulação denominado de chaveamento por deslocamento de fase por compensação de quadratura (O-QPSK), o que proporciona uma taxa de transmissão e recepção de 250 kbps no meio sem fio. Essa camada é responsável pela medição do nível de energia detectada (ED), pela estimativa da qualidade da conexão (LQI), por avaliar se o canal de comunicação utilizado está ou não livre e pela transmissão e recepção de bits de dados.

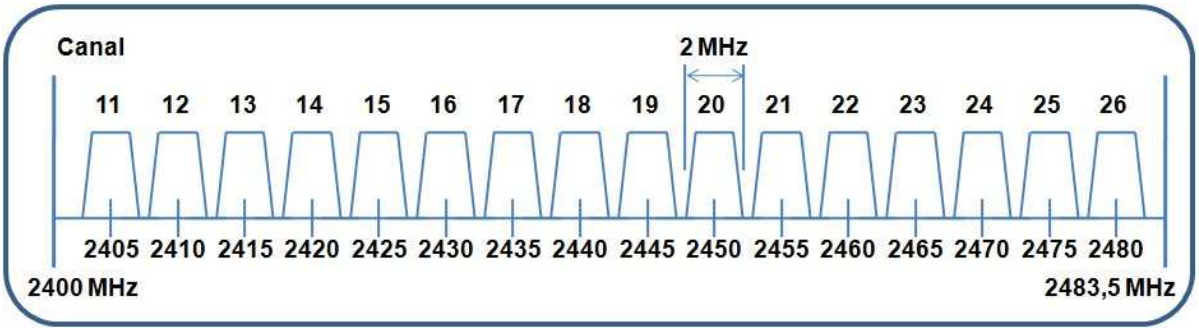


Figura 41 – Canal selecionado do padrão IEEE-802.15.4™-2006-LR-WPAN

As características da camada de controle de acesso ao meio (MAC) selecionadas foram a topologia estrela, na qual o dispositivo com funções completas (FFD) é o nó

sorvedouro e os dispositivos com funções reduzidas (RFD) são os nós sensores, identificados na figura 28. Essa topologia foi adotada em função da sua simplicidade, pois todos os elementos constituintes possuem uma conexão única entre o nó sensor e o nó sorvedouro, tornando a comunicação delimitada entre o nó coordenador e os dispositivos associados.

O presente trabalho não adotou o modo de operação que utiliza o mecanismo de compartimento de tempos garantidos (GTS) devido à limitação dessa funcionalidade de suportar no máximo sete dispositivos simultâneos, o que limita a sua utilização na prática. O acesso ao meio é feito por contenção, utilizando a técnica CSMA-CA, para evitar colisões de pacotes e sem definição de um compartimento de tempo específico (*unslotted*).

O tamanho da carga útil utilizada no pacote sonda do nó sorvedouro é de 19 bytes e no pacote sonda do nó sensor é de 31 bytes. O padrão IEEE 802.15.4™-2006 LR-WPAN limita o comprimento máximo da carga útil em 128 bytes, sendo que quantidades maiores de dados devem ser divididas em pacotes com no máximo este número de bytes. Essa limitação deve ser observada caso o pacote sonda ultrapasse o tamanho máximo permitido. O quadro 9 apresenta um resumo das características da Rede de Sensores Sem Fio utilizada no ambiente de testes.

Especificação	Seleção	Camada
Frequência de operação (Hz)	2450	Física (PHY)
Canal de operação	20	Física (PHY)
Técnica de modulação	DSSS/O-QPSK	Física (PHY)
Potência de transmissão (dBm)	+3	Física (PHY)
Topologia	Estrela	Controle de acesso ao meio (MAC)
Mecanismo de acesso ao meio	CSMA/CA	Controle de acesso ao meio (MAC)
Identificação da rede PAN	0xBEEF	Controle de acesso ao meio (MAC)
Carga útil do nó sensor (Bytes)	31	Controle de acesso ao meio (MAC)
Carga útil do nó sorvedouro (Bytes)	19	Controle de acesso ao meio (MAC)

Quadro 9 – Características da Rede de Sensores sem Fio utilizada no ambiente de testes

6.5 AMBIENTE DE TESTES

Esta seção apresenta o ambiente físico utilizado nos testes realizados para o levantamento do perfil dos aspectos temporais e para o levantamento da manutenção da qualidade da conexão resultantes da implantação da instrumentalização de software da Rede de Sensores Sem Fio.

6.5.1 Ambiente físico da realização dos testes

O ambiente físico utilizado para a realização dos testes foi selecionado em função de possuir uma área compatível com as especificações de uma rede pessoal, pela disponibilidade e facilidade de acesso. O laboratório de Engenharia Eletrônica da "Fundação Santo André" possui uma área de aproximadamente 88 m^2, com bancadas, armários e equipamentos. Os nós sensores foram posicionados no ambiente com diferentes distâncias em relação ao nó sorvedouro, conforme pode ser verificado na figura 42. Os nós sensores podem ser localizados na figura pelos identificadores Sen1, Sen2, Sen3 e Sen4 e o nó sorvedouro está identificado como Sor0, de acordo com a nomenclatura utilizada na figura 28. Os cinco nós foram montados em pedestais para que os mesmos ficassem localizados a uma distância de 1,5 m de altura em relação ao piso, o que proporciona uma melhor eficiência na propagação das ondas de rádio, em relação a sua colocação diretamente no solo. Todos os testes foram realizados nesse ambiente e em períodos em que não estava sendo utilizado pelos alunos. As distâncias aproximadas entre o nó sorvedouro e os nós sensores são: nó sensor 1 (9,80 m), nó sensor 2 (8,00 m), nó sensor 3 (7,40 m) e nó sensor 4 (4,10 m).

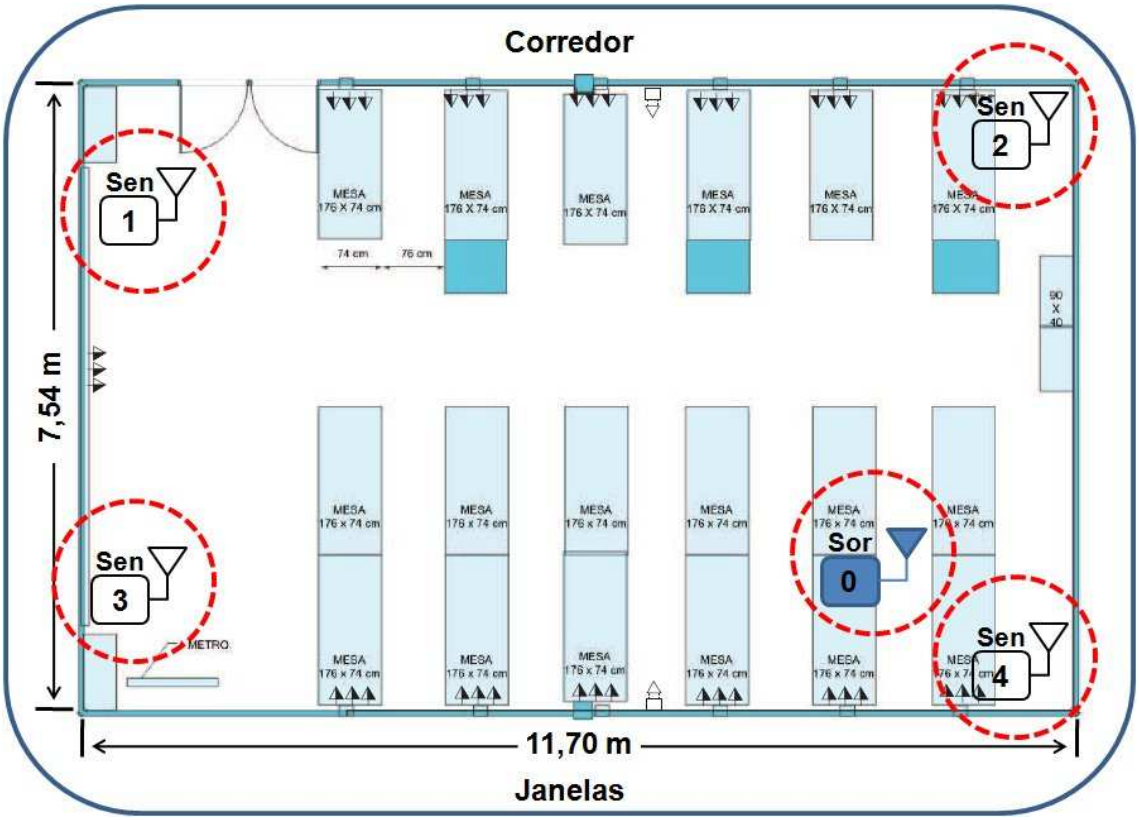


Figura 42 – Planta baixa do ambiente físico utilizado nos testes

6.6 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Este capítulo apresentou a descrição do ambiente de testes da proposta de utilização do conceito de instrumentalização de software, em Redes de Sensores Sem Fio, suportadas pelo padrão IEEE 802.15.4^{TM}-2006 LR-WPAN, em aplicações com restrições temporais.

Foram apresentadas a arquitetura do ambiente de testes, o instrumento de software, as características de hardware dos nós sensores e do nó sorvedouro, bem como a implantação da instrumentalização de software na Rede de Sensores sem Fio sob análise.

7 RESULTADOS DA INSTRUMENTALIZAÇÃO

Este capítulo apresenta, por meio dos resultados de testes realizados, o potencial representado pela instrumentalização proposta para Redes de Sensores Sem Fio. Conforme já destacado anteriormente, a rede seguiu a arquitetura descrita no capítulo 6, sendo que os objetivos da instrumentalização foram o levantamento do perfil desse tipo de rede em termos de aspectos temporais e manutenção da qualidade da conexão.

É importante destacar que o objetivo desta tese é propor a utilização do conceito de instrumentalização de software em Redes de Sensores Sem Fio. Desta forma os resultados aqui apresentados visam demonstrar a potencialidade da proposta e uma possível análise detalhada dos resultados apresentados não faz parte do escopo desta tese.

Com o objetivo de se levantar os perfis citados de forma mais abrangente, os testes foram realizados de forma individual e de forma coletiva. Neste ponto, é necessário esclarecer que a comunicação, em operação na forma individual, ocorre somente entre um determinado nó sensor e o nó sorvedouro, ficando os demais nós desligados. Nessa situação não ocorre a concorrência pelo acesso ao canal de comunicação imposta pelos demais nós sensores. Na forma coletiva a comunicação ocorre entre todos os quatro nós sensores e o nó sorvedouro, levando à necessidade da utilização dos mecanismos de controle de acesso ao canal de comunicação no sentido de se evitar colisões e perdas de pacotes.

7.1 RESULTADOS DA INSTRUMENTALIZAÇÃO NO LEVANTAMENTO DO PERFIL DOS ASPECTOS TEMPORAIS

A instrumentalização realizada permitiu o levantamento do perfil dos aspectos temporais da rede. A instrumentalização de software, conforme descrita na subseção 5.2.1 do capítulo 5 e desenvolvida de acordo com a figura 38 do capítulo

6, foi realizada por meio da alteração do software do nó sorvedouro para o registro do valor do relógio, no instante da transmissão (RelTxSrv), no pacote sonda sorvedouro (anexo-C). De forma análoga, o software do nó sensor foi alterado para incluir, no pacote sonda do nó sensor, o registro dos seguintes momentos: a) o instante da recepção do pacote sonda do sorvedouro (RelRxNsr); b) o instante da transmissão do pacote sonda nó sensor (RelTxNsr); e c) o instante da transmissão do pacote sonda sorvedouro (RelTxSrv), conforme pode ser verificado no anexo-D.

O nó sorvedouro, ao receber o pacote sonda do nó sensor, registra o instante dessa ocorrência (RelRxSrv), possibilitando o cálculo, pelo instrumento de software, do intervalo de tempo de ida-e-volta (RTT), como sendo o intervalo de tempo decorrido entre os valores de RelTxSrv e RelRxSrv.

Para o levantamento do perfil dos aspectos temporais, por meio da instrumentalização de software, foram utilizadas cinco configurações, com o intuito de se avaliar o perfil de cada nó, de forma individual e coletiva. Os testes foram realizados de acordo com o diagrama de relacionamento ilustrado na figura 28, no qual o nó sorvedouro, conforme os parâmetros configurados no instrumento de software pelo usuário, envia o pacote sonda para o nó sensor, o qual devolve um pacote sonda para o sorvedouro contendo os registros dos relógios mencionados.

O quadro 10 relaciona as características das configurações programadas. A configuração “A” seleciona o trajeto nó sorvedouro, nó sensor 1, retornando ao nó sorvedouro. Esse processo é repetido pelo número de vezes programado no instrumento de software e em intervalos de tempo de 5 segundos.

As demais configurações operam de maneira semelhante. Para o teste de levantamento dos aspectos temporais de forma coletiva, foi utilizada a configuração “E”. Essa configuração realiza a radiodifusão do pacote sonda pelo nó sorvedouro a todos os nós sensores da rede, os quais devem retransmitir o pacote sonda para o nó sorvedouro. Essa situação gera a necessidade da utilização de mecanismos de controle de acesso ao meio pelos nós sensores, para evitar colisões, e consequentemente, perdas de pacotes sonda transmitidos, tendo em vista que todos os nós tentam transmitir o pacote sonda, praticamente, ao mesmo tempo para o nó

sorvedouro. O instrumento de software criado possibilitou a verificação de todas essas configurações.

É importante destacar que o intervalo entre os envios e a quantidade de pacotes sonda trocados entre o nó sorvedouro e os nós sensores foram configurados por meio do instrumento de software, conforme destacado na figura 30 para cada configuração. A escolha desses parâmetros levou em consideração que, para o levantamento do perfil é recomendável uma quantidade significativa de pacotes sonda, sendo considerado como número mínimo a ser utilizado neste trabalho, a quantidade de 3.000 pacotes.

Configuração	**Origem Tx / Destino Rx nó**	**Intermediário Rx/Tx nó sensor**	**Número pacotes**	**Intervalo (mseg)**
A	sorvedouro	1	3.600	5.000
B	sorvedouro	2	3.600	5.000
C	sorvedouro	3	3.600	5.000
D	sorvedouro	4	3.600	5.000
E	sorvedouro	Radiodifusão	3.600	5.000

Quadro 10 – Configurações utilizadas no levantamento do perfil dos aspectos temporais

7.1.1 Instrumentalização dos Aspectos Temporais

Esta subseção apresenta os resultados obtidos, por meio do instrumento de software citado, no levantamento do perfil dos aspectos temporais da Rede de Sensores Sem Fio. Para uma melhor apresentação e análise dos resultados obtidos, foi realizada a divisão dos dados coletados de forma individual e coletiva.

Para a avaliação do resultado, além das informações disponibilizadas pelo instrumento de software, dos valores de RTT atuais, médios, mínimos e máximos, conforme destacados na figura 31, também foi realizada a análise gráfica dos dados em termos de valores de RTT obtidos, da distribuição de frequência e também diagrama de barras dos valores mínimo, médio e máximo levantados de forma

individual e coletiva, no intuito de evidenciar as possibilidades de análise a serem adicionadas ao conceito de instrumentalização de software.

7.1.1.1 Instrumentalização dos Aspectos Temporais Individuais

O quadro 11 apresenta o resumo dos valores obtidos no levantamento dos aspectos temporais individuais de cada nó sensor constituinte da Rede de Sensores Sem Fio. O quadro relaciona o número de pacotes utilizado, os valores mínimo, médio, máximo obtidos e o cálculo da amplitude (diferença entre o valor máximo e mínimo).

Aspectos temporais – Individual				
Nó sensor	**Nó 1**	**Nó 2**	**Nó 3**	**Nó 4**
Número de pacotes	3.600	3.600	3.600	3.600
Rtt-mínimo[us]	83.104	81.952	83.184	81.968
Rtt-médio[us]	85.426	84.424	85.516	84.397
Rtt-máximo[us]	87.632	109.440	87.728	86.800
Amplitude[us]	4.528	27.488	4.544	4.832

Quadro 11– Valores obtidos no levantamento do perfil dos aspectos temporais de forma individual

7.1.1.2 Instrumentalização dos Aspectos Temporais Coletivos

O quadro 12 resume os valores obtidos no levantamento dos aspectos temporais da Rede de Sensores Sem Fio de forma coletiva. O quadro apresenta o número de pacotes utilizado, os valores mínimo, médio, máximo obtidos e o cálculo da amplitude.

Aspectos temporais – Coletivo					
Nó sensor	**Nó 1**	**Nó 2**	**Nó 3**	**Nó 4**	**Geral**
Número de pacotes	3.600	3.600	3.600	3.600	14.400
Rtt-mínimo[us]	83.120	81.984	83.232	82.656	81.984
Rtt-médio[us]	347.845	200.658	346.439	310.303	301.311
Rtt-máximo[us]	571.488	571.408	571.632	571.888	571.888
Amplitude[us]	488.368	489.424	488.400	489.232	489.904

Quadro 12 – Valores obtidos no levantamento do perfil dos aspectos temporais de forma coletiva

7.1.1.3 Análise Gráfica dos Aspectos Temporais

Utilizando a base de dados fornecida pela instrumentalização de software é possível a análise de algumas características adicionais, não visíveis na tela do instrumento disponível ao usuário.

A construção do gráfico dos valores do tempo de ida-e-volta (RTT), destacando os valores mínimos, médios e máximos, auxilia na compreensão da variação do tempo de resposta em cada uma das configurações ensaiadas. Essa funcionalidade, incorporada ao instrumento de software, reduz os esforços utilizados na análise dos dados coletados.

Nesse sentido, a figura 43 representa os gráficos dos valores de RTT obtidos de forma individual para o nó sensor 1. Para o levantamento do perfil desejado de forma individual, ou seja, para cada nó sensor operando sozinho, sem influência dos demais nós sensores da rede, pode-se notar uma variação homogênea entre os valores mínimo e máximo no decorrer das medições. Vale ressaltar que os resultados são semelhantes para os nós sensores 2, 3 e 4.

Por meio do quadro 11 pode-se verificar que o tempo-de-ida e volta (RTT) para o nó sensor 1 ficou situado entre 83.104 e 87.632 microsegundos, com valor médio de 85.426 microsegundos, conforme destacado na figura 43.

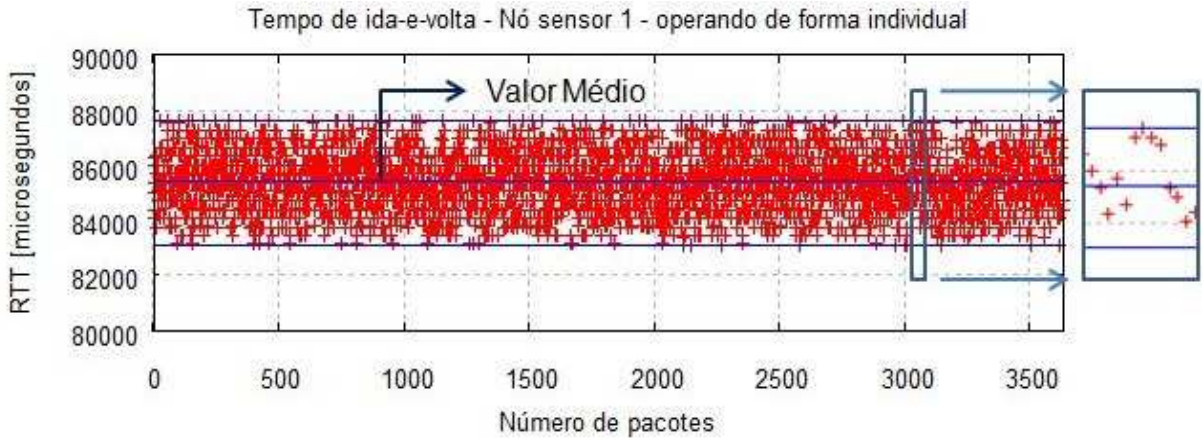


Figura 43 – Valores do tempo de ida-e-volta obtidos de forma individual

No levantamento do perfil desejado de forma coletiva, conforme gráfico da figura 44, nota-se a estratificação dos valores obtidos em 4 faixas bem definidas. Isso se deve à operação simultânea dos 4 nós sensores constituintes da Rede de Sensores Sem Fio utilizada nos ensaios. Vale ressaltar que os resultados são semelhantes para os nós sensores 2, 3 e 4.

Na operação do nó sensor 1 de forma coletiva, por meio do quadro 12, pode-se verificar que o tempo de ida-e-volta (RTT) ficou situado entre 83.120 e 571.488 microsegundos, com um valor médio de 347.845 microsegundos, conforme destacado na figura 44. É interessante notar que, embora a variação do RTT seja muito maior do que quando operando de forma individual, os valores obtidos por meio da instrumentalização de software ficaram concentrados em torno de 4 faixas de valores bem definidas.

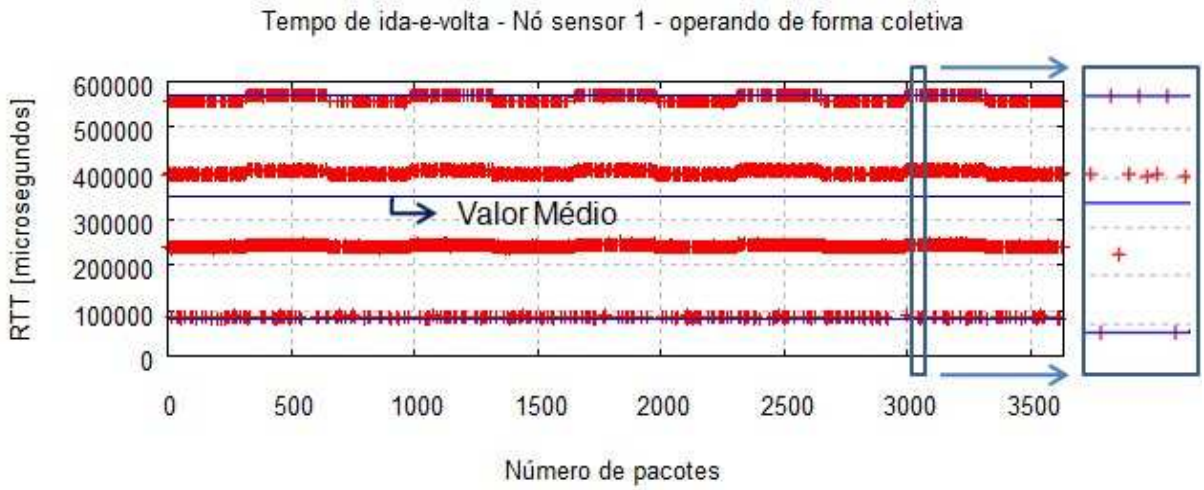


Figura 44 – Valores do tempo de ida-e-volta obtidos de forma coletiva

A figura 45 mostra o gráfico da análise da distribuição de frequência do nó sensor 1 operando de forma individual. Este gráfico mostra o número de pacotes que apresentam o mesmo valor para o tempo de ida-e-volta (RTT) destacado como frequência no gráfico. Nessa figura é possível notar uma distribuição balanceada em torno da média e com raias equidistantes em termos de tempo de ida-e-volta (RTT).

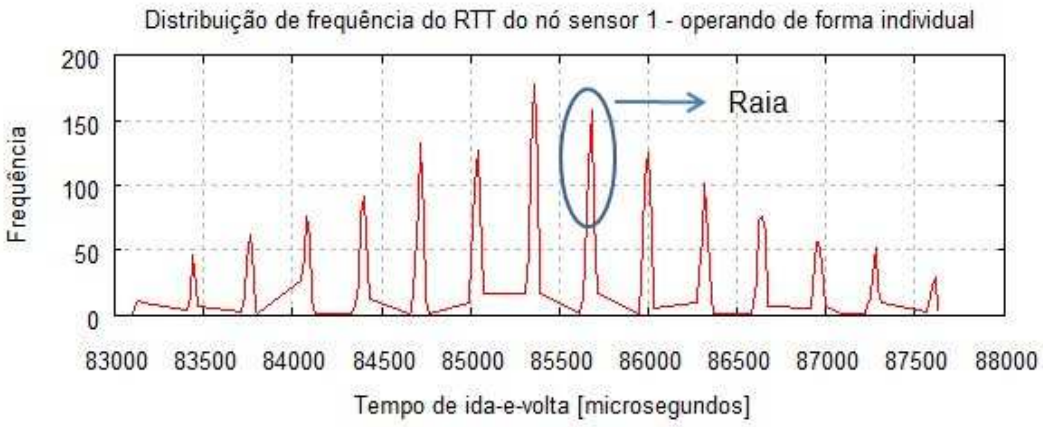


Figura 45 – Distribuição de frequência do tempo de ida-e-volta do nó sensor 1 operando de forma individual

O gráfico da figura 46 apresenta o comparativo dos valores mínimos, médios e máximos obtidos em operação individual e coletiva devido ao nó sensor 1. Por meio da construção do diagrama de barras é possível verificar que o valor do tempo de ida-e-volta (RTT) aumenta significativamente quando o nó sensor 1 opera de forma coletiva e é um indicativo significativo na avaliação do tempo de resposta da rede.

Pode-se observar nessa figura e com base nos quadros 11 e 12, que os valores de RTT mínimos, tanto nos ensaios individuais quanto nos coletivos, apresentam uma variação máxima de 1.246 microsegundos, enquanto que os valores máximos de RTT coletivo apresentam uma variação máxima de 480 microsegundos.

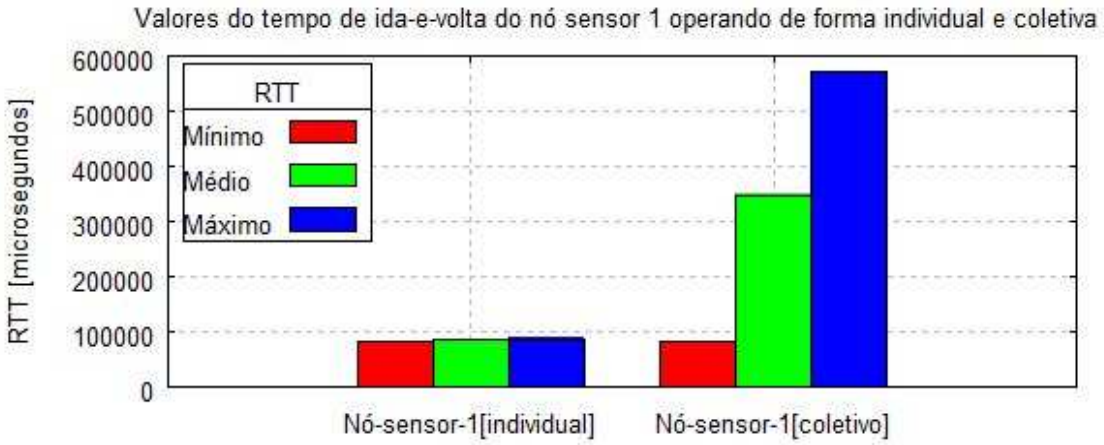


Figura 46 – Diagrama de barras comparativo do nó sensor 1 operando de forma individual e coletiva

7.1.1.4 Comentários sobre o perfil dos Aspectos temporais

Os resultados obtidos pela instrumentalização de software com o objetivo do levantamento do perfil dos aspectos temporais da Rede de Sensores Sem Fio foram satisfatórios e possibilitaram uma visão ampliada dos tempos de respostas envolvidos na operação da rede. A instrumentalização da rede possibilitou o levantamento desses valores e uma melhor compreensão do funcionamento da rede em termos temporais. A análise gráfica amplia e facilita a visão desses parâmetros de interesse.

7.2 RESULTADOS DA INSTRUMENTALIZAÇÃO NO LEVANTAMENTO DO PERFIL DA MANUTENÇÃO DA QUALIDADE DA CONEXÃO

A instrumentalização de software realizada possibilitou o levantamento do perfil da manutenção da qualidade da conexão da Rede de Sensores Sem Fio ensaiada. A instrumentalização, conforme descrita na subseção 5.2.1 do capítulo 5 e desenvolvida de acordo com a figura 38 do capítulo 6, foi realizada por meio das alterações do software do nó sorvedouro e do nó sensor.

As alterações do software do nó sorvedouro possibilitaram a configuração dos parâmetros de funcionamento da rede, ou seja, o número de pacotes a serem enviados pelos nós sensores ao sorvedouro e o intervalo de tempo entre envios. Ao receber os pacotes sonda enviados pelo nó sensor, o sorvedouro preenche o registro de dados, conforme relacionado na figura 23, e o envia ao instrumento de software via comunicação serial. As alterações podem ser verificadas no anexo-C.

As alterações para a instrumentalização do nó sensor foram efetuadas no sentido configurar o nó sensor para enviar pacotes sonda para o nó sorvedouro de acordo com o número de pacotes e intervalos de tempos programados. As informações constantes do quadro de dados sonda do nó sensor estão relacionados na figura 19 e possibilitaram o levantamento do perfil da manutenção da qualidade da conexão

Para o levantamento do perfil da manutenção da qualidade da conexão por meio da instrumentalização de software foram utilizadas cinco configurações, com o intuito de se avaliar o perfil de cada nó, de forma individual e coletiva. O mecanismo dos testes foi desenvolvido de acordo com a figura 28, na qual o usuário, por meio do instrumento de software, configura o teste desejado, selecionando o tipo de teste, o número de pacotes sonda transmitidos pelo nó sensor e o intervalo de tempo a ser observado entre as transmissões.

O quadro 13 relaciona as características das configurações utilizadas. A configuração “A” seleciona o nó sensor 1 para transmitir o número de pacotes sonda programados pelo usuário para o sorvedouro em intervalos de tempo de 5 segundos, e assim sucessivamente para as outras configurações.

Configuração	**Origem Tx Nó sensor**	**Destino Rx Nó**	**Número pacotes**	**Intervalo (mseg)**
A	1	sorvedouro	8.300	5.000
B	2	sorvedouro	8.300	5.000
C	3	sorvedouro	8.300	5.000
D	4	sorvedouro	8.300	5.000
E	1,2,3 e 4	sorvedouro	8.300	5.000

Quadro 13 – Configurações utilizadas no levantamento do perfil da manutenção da qualidade da conexão

7.2.1 Instrumentalização para Verificação da Qualidade da Conexão

Esta subseção apresenta os resultados obtidos, por meio do instrumento de software citado, no levantamento do perfil da manutenção da qualidade da conexão da Rede de Sensores Sem Fio. De forma análoga a seção anterior, foi realizada a divisão dos dados coletados de forma individual e coletiva para uma melhor apresentação e análise dos resultados obtidos.

7.2.1.1 Instrumentalização para Verificação da Qualidade da Conexão Individual

A seguir são apresentados os resultados obtidos pela instrumentalização da Rede de Sensores Sem Fio referente ao levantamento do perfil da manutenção da qualidade de conexão.

O quadro 14 apresenta os valores obtidos pelo nó sorvedouro referentes ao envio sequencial do pacote sonda por cada nó sensor constituinte da Rede de Sensores Sem Fio de forma individual. O quadro apresenta as quantidades de pacotes, os valores mínimos, médios e máximos obtidos e do cálculo da amplitude para a energia detectada (ED), a qualidade da conexão (LQI), a tensão da bateria (TB) e a temperatura de operação (TO). Para a taxa de recepção de pacotes (PRR) é indicado o percentual de pacotes efetivamente recebidos pelo nó sorvedouro. Os dados relacionados foram obtidos de acordo com a figura 32.

Manutenção da qualidade da conexão - Individual				
Nó sensor	**Nó 1**	**Nó 2**	**Nó 3**	**Nó 4**
Número de pacotes	8.300	8.300	8.300	8.300
ED-mínimo[dBm]	-76	-68	-63	-60
ED-médio[dBm]	-59,15	-57,91	-50,25	-51,00
ED-máximo[dBm]	-34	-37	-48	-36
Amplitude[dBm]	42	31	15	24
LQI-mínimo	59	93	114	127
LQI-médio	130,42	136,16	168,75	165,01
LQI-máximo	238	225	178	229
Amplitude	179	132	64	102
PRR[%]	100	100	100	100
TB-mínimo[mV]	2.550	2.550	2.550	2.550
TB-médio[mV]	2.588,36	2.550,41	2.577,27	2.569,88
TB-máximo[mV]	2.700	2.625	2.700	2.700
Amplitude[mV]	150	75	150	150
TO-mínimo[℃]	24	21	18	18
TO-médio[℃]	24,39	22,52	22,65	20,02
TO-máximo[℃]	26	25	44	22
Amplitude[℃]	2	4	26	4

Quadro 14 – Valores obtidos no levantamento do perfil da qualidade da conexão de forma individual

7.2.1.2 Instrumentalização para Verificação da Qualidade da Conexão Coletiva

A seguir são apresentados os resultados obtidos por meio da instrumentalização da Rede de Sensores Sem Fio, referente ao levantamento do perfil da manutenção da qualidade de conexão de forma coletiva. O quadro 15 apresenta as mesmas informações do quadro anterior para os sensores operando de forma coletiva.

Manutenção da qualidade da conexão - Coletivo					
Nó sensor	**Nó 1**	**Nó 2**	**Nó 3**	**Nó 4**	**Todos**
Número de pacotes	8.300	8.300	8.300	8.300	33.200
ED-mínimo[dBm]	-78	-76	-78	-78	-78
ED-médio[dBm]	-59,74	-67,84	-54,70	-58,37	-60,16
ED-máximo[dBm]	-38	-38	-38	-38	-38
Amplitude[dBm]	40	38	40	40	40
LQI-mínimo	51	59	85	42	42
LQI-médio	127,72	93,90	150,01	134,16	126,45
LQI-máximo	221	216	221	204	221
Amplitude	170	157	136	162	179
PRR[%]	99,988	100	100	99,976	99,991
TB-mínimo[mV]	2550	2550	2550	2550	2550
TB-médio[mV]	2.564,94	2.550,00	2.582,19	2.581,06	2.569,55
TB-máximo[mV]	2700	2550	2700	2700	2700
Amplitude[mV]	150	0	150	150	150
TO-mínimo[℃]	19	19	20	18	18
TO-médio[℃]	20,97	21,07	24,87	20,20	21,78
TO-máximo[℃]	22	22	41	22	41
Amplitude[℃]	3	3	21	4	23

Quadro 15 – Valores obtidos no levantamento do perfil coletivo dos aspectos da manutenção da qualidade da conexão

7.2.1.3 Análise gráfica dos aspectos da manutenção da qualidade da conexão

Essa análise foi realizada para cada uma das métricas envolvidas na avaliação da manutenção da qualidade da conexão, ou seja, a energia detectada (ED), o indicador da qualidade da conexão (LQI), a taxa de recepção de pacotes (PRR), a tensão da bateria (TB) e a temperatura de operação (TO). A seguir são apresentados os resultados da análise, de forma individual e também coletiva.

a) Análise gráfica da energia detectada (ED)

A figura 47 representa os gráficos obtidos por meio dos valores originados da instrumentalização da Rede de Sensores Sem Fio, da energia detectada obtidos de forma individual e coletiva para o nó sensor 1. Pela análise dessa figura verifica-se, por meio da quantidade de picos, a ocorrência de ligeiras variações da energia detectada (ED), as quais são mais acentuadas quando o nó sensor está operando de forma coletiva. Os gráficos mostram que o nó sensor 1 manteve a operação em torno de -60 dBm em ambas formas de operação. No intuito de proporcionar uma melhor compreensão do gráfico, foi destacada a ocorrência de um pico e também a ampliação de uma faixa de valores.

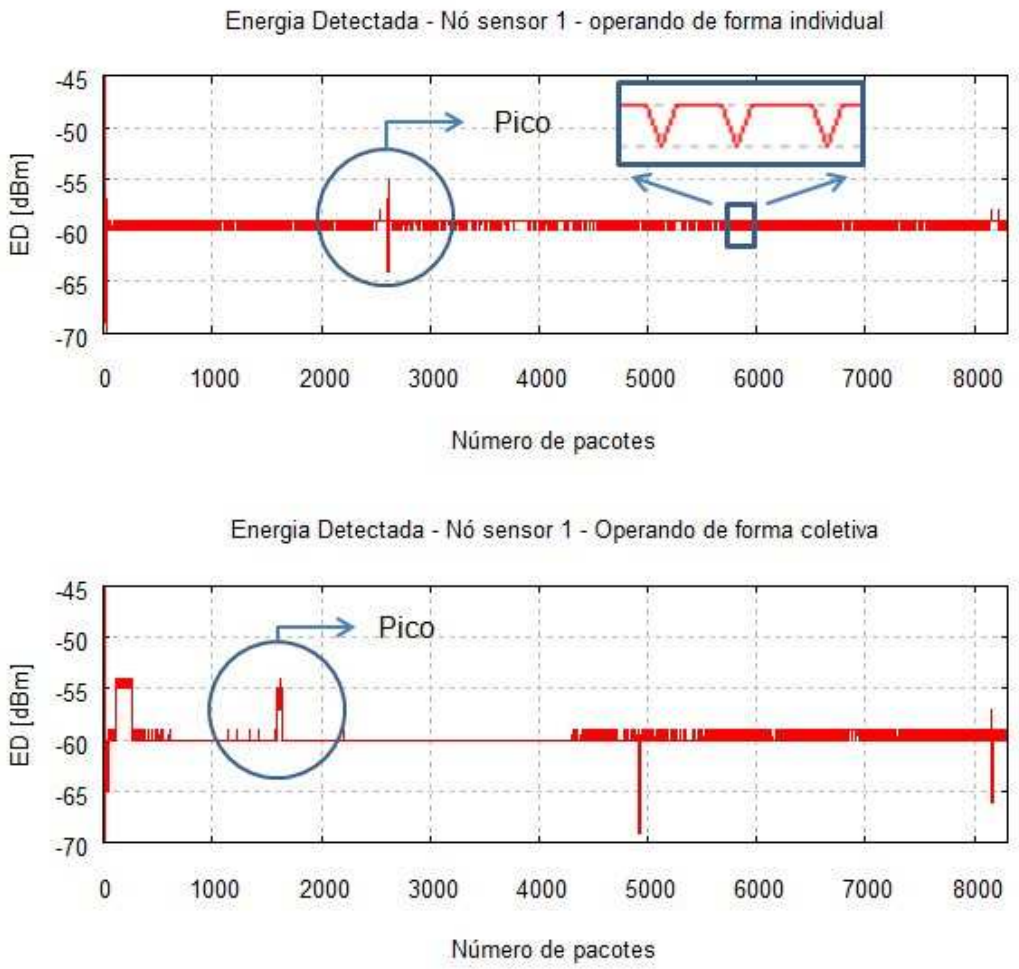


Figura 47 – Valores da energia detectada (ED) obtidos de forma individual e coletiva

Pela análise dos quadros 14 e 15, por meio dos valores de ED médio, verifica-se que o nó sensor 1 apresentou uma leve queda de 0,59 dBm, o nó sensor 2 apresentou uma queda de 9,93 dBm, o nó sensor 3 apresentou uma queda de 4,45 dBm e o nó sensor 4 apresentou uma queda de 7,37 dBm quando em operação de forma coletiva. Os valores apresentados pelos nós sensores 1 e 2, mais distantes do nó sorvedouro, conforme figura 42, apresentaram um nível de energia detectada (ED) mais baixo do que os nós sensores 3 e 4 que estão mais próximos do nó sorvedouro. Nota-se também que houve uma queda geral na energia detectada (ED) quando os nós sensores operam de forma coletiva.

b) Análise gráfica do indicador da qualidade da conexão (LQI)

A figura 48 representa os gráficos obtidos por meio dos valores originados da instrumentalização da Rede de Sensores Sem Fio, da qualidade da conexão (LQI) obtidos de forma individual e coletiva para o nó sensor 1. O valor do LQI é calculado com base na energia detectada (ED). O nós sensores 2, 3 e 4 apresentaram um comportamento semelhante.

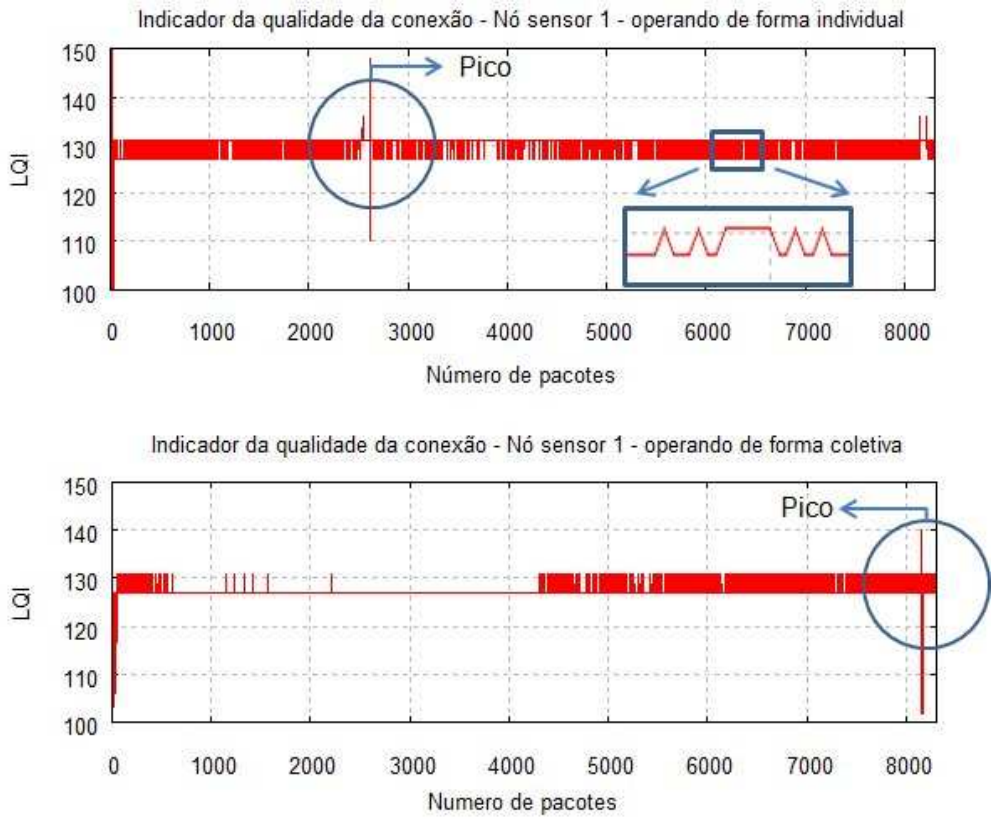


Figura 48 – Valores do indicador da qualidade da conexão (LQI) obtidos de forma individual e coletiva

c) Análise gráfica da taxa de recepção de pacotes (PRR)

A figura 49 representa os gráficos obtidos por meio dos valores originados da instrumentalização da Rede de Sensores Sem Fio, da taxa de recepção de pacotes (PRR) obtidos de forma individual e coletiva para o nó sensor 1. Para a construção desse gráfico foi utilizado o valor 1 para as situações em que o pacote sonda foi efetivamente recebido pelo nó sorvedouro e o valor 0 para as situações em que o pacote sonda não foi recebido pelo nó sorvedouro, sinalizando a ocorrência de perda de pacote.

Pode-se notar pela figura que não houve perda de nenhum pacote quando o nó 1 operou de forma individual. Nesse ensaio o nó sensor 1 apresentou a perda de 1 pacote em uma amostra de 8.300 pacotes quando operou de forma coletiva. Os resultados obtidos para os demais nós sensores podem ver verificados por meio dos quadros 14 e 15.

Figura 49 – Valores da taxa de recepção de pacotes (PRR) obtidos de forma individual e coletiva

d) Análise gráfica da tensão da bateria (TB)

A figura 50 representa os gráficos obtidos por meio dos valores originados da instrumentalização da Rede de Sensores Sem Fio, da tensão da bateria (TB) obtidos de forma individual e coletiva para o nó sensor 1. As baterias utilizadas nos ensaios são recarregáveis e por meio da figura verifica-se que as mesmas possuem um comportamento mais estável com tensões em torno de 2.550 mV, sendo que para tensões superiores a descarga das mesmas é mais acentuada. O algoritmo de software que fornece a tensão da bateria trabalha com passos de 75 mV para tensões iguais ou superiores a 2.550 mV e de 50 mV para tensões abaixo desse valor. Essa característica justifica as variações abruptas observadas entre 2.550 e 2.700 mV. Para os nós sensores 2, 3 e 4 os resultados foram semelhantes e podem ser verificados por meio dos quadros 14 e 15.

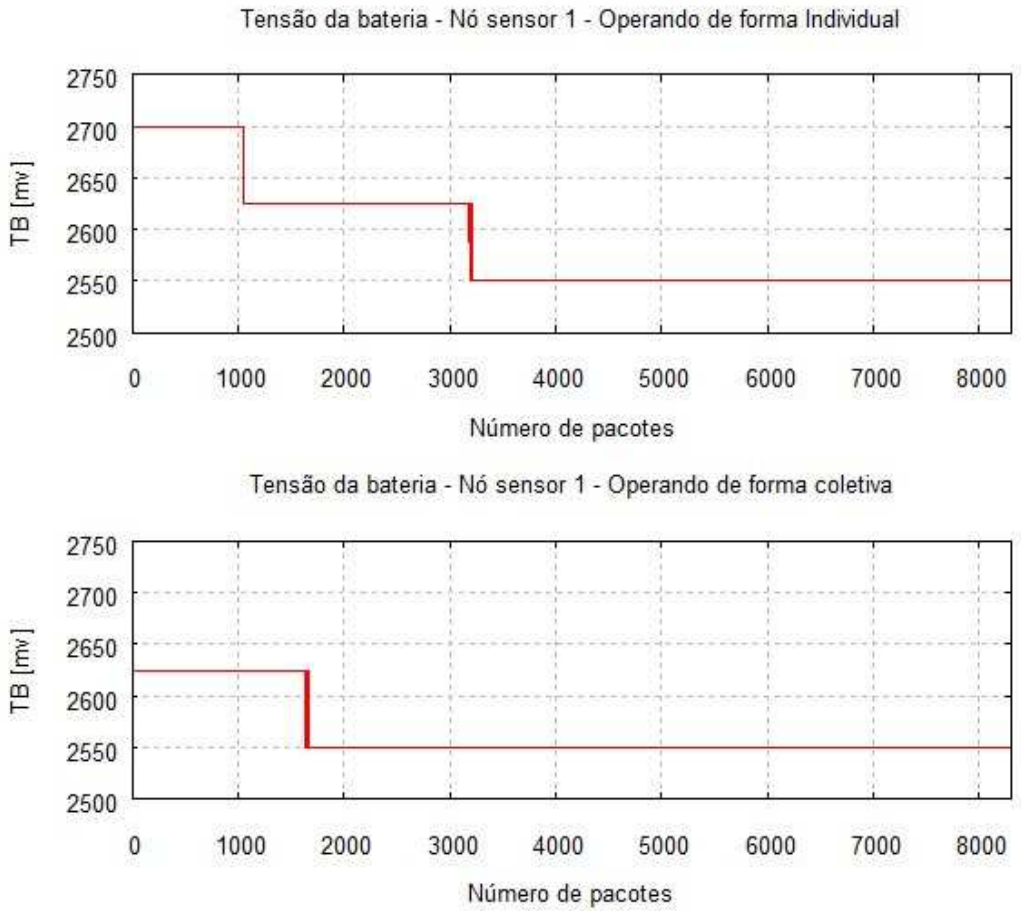


Figura 50 – Valores da tensão da bateria (TB) obtidos de forma individual e coletiva

e) Análise gráfica da temperatura de operação (TO)

A figura 51 representa os gráficos obtidos por meio dos valores originados da instrumentalização da Rede de Sensores Sem Fio, da temperatura de operação (TO) obtidos de forma individual e coletiva para o nó sensor 3. A análise desses gráficos possibilitou o conhecimento do comportamento da temperatura de operação de forma individual e coletiva.

O nó sensor 3 foi escolhido para a análise gráfica pois apresentou variações de temperatura de operação significativas. Essas variações foram devidas a localização do nó sensor 3 próximo a janela com incidência direta do sol nesse período. Os nós sensores 1 e 2 estavam posicionados ao lado de um corredor interno e o nó sensor 4 também estava próximo a janela, porém esta região estava protegida da incidência direta do sol, conforme figura 42.

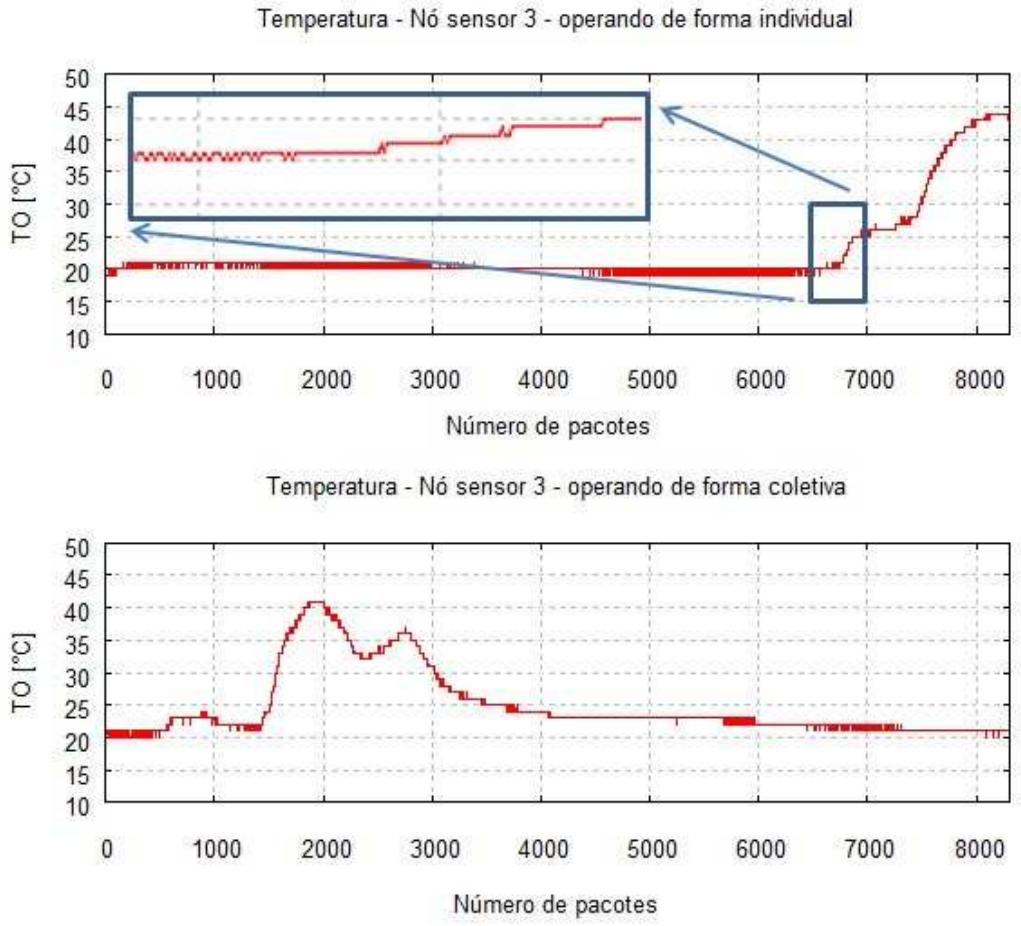


Figura 51 – Valores da temperatura de operação (TO) obtidos de forma individual e coletiva

7.2.1.4 Comentários sobre o perfil da manutenção da qualidade da conexão

Os resultados obtidos, por meio da instrumentalização de software da Rede de Sensores Sem Fio, no levantamento do perfil da manutenção da qualidade da conexão foram satisfatórios e possibilitaram uma visão ampliada da operação da rede, proporcionando dessa forma o conhecimento do comportamento da rede em termos da manutenção da qualidade da conexão. A análise gráfica amplia e facilita a visão dos parâmetros de interesse.

7.3 CONSIDERAÇÕES FINAIS DESTE CAPÍTULO

Neste capítulo foram apresentados os testes realizados e os resultados obtidos com o intuito do levantamento dos perfis dos aspectos temporais e da manutenção da qualidade da conexão. A análise gráfica dos dados obtidos proporcionou uma visão mais consistente do comportamento da Rede de Sensores Sem fio de ambos os perfis.

8 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as contribuições, conclusões e sugestões para trabalhos futuros na área de Redes de Sensores Sem Fio envolvendo instrumentalização de software.

8.1 CONTRIBUIÇÕES E CONCLUSÕES

A principal contribuição desta pesquisa é a proposta da utilização do conceito de instrumentalização de software aplicada a Redes de Sensores Sem Fio, principalmente em aplicações que possuam restrições temporais e que necessitem da manutenção da qualidade da conexão para seu adequado funcionamento. A instrumentalização de software possibilita a monitoração automática desses tipos de redes, e consequentemente o conhecimento mais detalhado de sua operação.

O desenvolvimento de uma Rede de Sensores Sem Fio utilizando os conceitos de instrumentalização de software, abordando principalmente características relacionadas aos aspectos temporais e à manutenção da qualidade de conexão comprovou, conforme os resultados obtidos e relacionados no capítulo 7, que a utilização desta proposta é uma ferramenta importante e de grande valia no estudo desse tipo de redes.

Vale ressaltar que a utilização do conceito de instrumentalização de software em Redes de Sensores Sem Fio possibilita a criação de ferramentas específicas para o estudo do desempenho dessas redes em termos do desenvolvimento ou estudo de algoritmos de roteamento, de algoritmos de controle de acesso ao meio e de aplicações com restrições de algum determinado recurso, entre outros. Essa nova classe de instrumentos amplia o leque de oportunidades para o efetivo conhecimento do comportamento operacional desses tipos de redes.

É importante destacar que por meio do levantamento do perfil dos aspectos temporais da rede é possível verificar se a Rede de Sensores Sem Fio está ou não adequada para ser utilizada no desenvolvimento de sistemas de tempo real.

Como regra geral, espera-se que na maioria dos casos a instrumentalização de software não é recomendada para sistemas nos quais a violação do tempo máximo de resposta possa ter consequências danosas, tanto para o ambiente quanto para as pessoas. Por outro lado foi constatado que a rede utilizada pode ser aplicada em situações em que a violação esporádica do tempo seja aceitável.

A monitoração do desempenho das Redes de Sensores Sem Fio é uma tarefa complexa, dependente da aplicação e que uma das melhores formas de se desenvolver mecanismos para o acompanhamento automático desse desempenho é por meio da instrumentalização de software dessas redes.

O presente trabalho proporcionou uma visão consistente do comportamento da Rede de Sensores Sem Fio tanto em termos de aspectos temporais quanto da manutenção da qualidade da conexão. O conceito é amplo e não se restringe a uma aplicação específica, podendo ser utilizado em diversas oportunidades em que seja necessário o conhecimento e monitoração do desempenho da rede.

A instrumentalização de software é um processo de adição, ou seja, de complementar um software aplicativo com rotinas que possibilitem a instrumentalização da rede. Nesse sentido deve-se levar em conta o aumento no consumo dos recursos devido a esse processo.

O esforço empregado na realização do presente trabalho tem o objetivo de reduzir a deficiência de ferramentas utilizadas na avaliação das características de interesse de Redes de Sensores Sem Fio e que seja utilizado como base no desenvolvimento e estudos desse tipo de tecnologia.

8.2 TRABALHOS FUTUROS

A aplicação do conceito de instrumentalização de software em Redes de Sensores Sem Fio é uma área vasta para o desenvolvimento de aplicações diversas. A incorporação ao instrumento de software da análise gráfica, desenvolvida no presente trabalho, possibilita uma visão mais ampla, do comportamento operacional da rede pelo usuário. A construção de gráficos de tendência, histogramas e a verificação da existência de determinadas correlações devido às variações temporais e espaciais, de forma automática, ampliará o conhecimento do comportamento da rede.

Como mencionado em vários pontos desse trabalho o consumo de energia é uma tônica presente em praticamente todos os estudos relacionados a Redes de Sensores Sem Fio. Nesse sentido a pesquisa da variação no consumo de energia em função da instrumentalização de software dessas redes é importante.

Algumas condições de contorno consideradas na arquitetura proposta, como a topologia em estrela e o método de acesso ao meio baseado na tecnologia CSMA-CA, sem definição do compartimento de tempo (*unslotted*), permitem que sejam realizadas novas propostas de pesquisas. O estudo do comportamento de Redes de Sensores Sem Fio, utilizando os conceitos de instrumentalização de software, operando com uma topologia ponto-a-ponto, na qual as mensagens trocadas na rede passam por vários nós sensores antes de chegar ao destino é um desafio a ser equacionado. Esse estudo é justificado pelo número significativo de redes que adotam essa topologia.

Outro campo de pesquisa é o comportamento das Redes de Sensores Sem Fio operando com compartimento de tempo garantido (GTS), no qual o controle de acesso ao meio é assegurado por esse mecanismo. A instrumentalização de software nessas condições permitirá a monitoração dos atrasos temporais especificados por essa modalidade de operação.

A instrumentalização de software poderá ser ampliada por meio da inclusão de conceitos de automação industrial, com a integração de softwares supervisórios, que possibilitem além da aquisição de dados e da monitoração dos parâmetros de interesse da rede, a execução de algoritmos de controle de alarmes, sinalizando ao operador a ocorrência de situações indesejadas.

9 REFERÊNCIAS

AKYLDIZ, I. F. et al. **Wireless sensor networks: a survey**. Computer Networks. New York, NY, USA, v. 38, p. 393–422, 2002.

ALVES, M. et al. **Quality-of-service in wireless sensor networks: state-of-the-art and future directions**. CISTER/ISEP Polytechnic Institute of Porto, Portugal. Nov 2009.

ATMEL. **8-bit avr microcontroller with low power 2.4ghz transceiver for Zigbee and IEEE 802.15.4-atmega128rfa1.** 2009.

ATMEL. **AVR2025: IEEE 802.15.4 mac software package – user guide. Application note preliminary**. 2010.

BARONTI, P.et al. **Wireless sensor networks: a survey on the state of the art and the 802.15.4 and Zigbee standards**. Newton, USA. Computer Communications, v. 30, p. 1655–1695, May 2007.

BEUTEL, J. et al. **Deployment techniques for sensor networks**. Chapter of Sensor Networks – Where the Theory Meets Practice, p. 219–248, Springer, 2010.

BURGESS, S. et al. **Harnessing wireless sensor technologies to advance forest ecology and agricultural research**. Elsevier B. V. Agricultural and Forest Meteorology, v. 150, n. 1, p. 30-37, 2010.

CANSADO, J. C. A. et al. Data farming quality, improving the wsn safety. In: **Reliability, Risk and Safety Theory and Application, ESREL 2009**, (London), p. 847–852, CRC Press, 2010.

CHITNIS, M.et al. Impact of the operating system on the qos offered by an IEEE 802.15.4-compliant sensor network. In: **7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems.** Toulouse, France - Nov. 2007.

CULLER, D.; ESTRIN, D.; SRIVASTAVA, M. **Guest editors' introduction: overview of sensor networks**. Computer, v. 37, n. 8, p. 41–49, 2004.

GRAHAM, S. L.; KESSLER P. B.; MCKUSICK, M. K. Gprof: a Call Graph Execution Profiler. In: **SIGPLAN '82 Proceedings of the 1982 SIGPLAN symposium on Compiler construction.** New York, June1982.

GRATTON, D. A. **Developing practical wireless applications**. Burlington, MA, USA: Elsevier Inc. Digital Press, 2007.

HAMMAD M.; COOK J. Lightweight Deployable Software Monitoring for Sensor Networks. **In: ICCCN '09 Proceedings of the 2009 Proceedings of 18th International Conference on Computer Communications and Networks. IEEE Computer Society**, Washington, 2009.

IEEE. **Standard dictionary of electrical and electronics terms**. ANSI/IEEE Std 100-1984. 3 ed. New York, NY, USA. 1984.

_____.**Part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans)**. **Standard for information technology – telecommunication and information exchange between systems – local and metropolitan area networks – specific requirements**. IEEE Computer Society, New York, NY, USA, Sept. 2006.

JUANG, P. et al. **Energy efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet**. New York. SIGPLAN Not., v. 37, p. 96–107, Oct. 2002.

JURCIK, P. et al. A simulation model for the IEEE 802.15.4 protocol: Delay/throughput evaluation of the gts mechanism. Istambul. In: **Modeling, Analysis, and Simulation of Computer and Telecommunication Systems,2007. MASCOTS '07. 15th International Symposium on**, p. 109 –116, Oct. 2007.

KARL, H.; WILLIG, A. **A short survey of wireless sensor networks**. Technical University Berlin. Tech. Rep.TKN-03-018, Telecommunication Networks Group, Oct. 2003.

KARL, H.; WILLIG, A. **Protocols and Architectures for Wireless Sensor Networks**. Chichester, England. John Wiley & Sons, 2005.

LEE, Y.-D.; JEONG, D.-U.; LEE, H.-J. Performance analysis of wireless link quality in wireless sensor networks. Seoul. In: **Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on**, p. 1006–1010, Dec. 2010.

LEVON, J. **Profiling in linux howto**. Report, 2002. Disponível em: <http://www.movementarian.org/docs/linux-profiling/>. Acesso em: 18 de jul. 2011.

LI, Y.; THAI, M. T.; WU, W. **Wireless sensor networks and applications**. New York, NY, USA. Springer Science+Business Media, LLC, 2008.

MAGALHÃES, I. L.; PINHEIRO, W. B**. Gerenciamento de serviços de ti na prática – uma abordagem com base na itil**. São Paulo: Novatec Editora, 2010.

MERRET, G. V. et al. Energy-aware simulations for wireless sensor networks. In: **SECON ’09: Proceedings of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks**. Rome, Italy, p. 64-71, IEEE, 2009.

MICROSOFT. Usando o comando ping. Windows Server TechCenter. Disponível em: < http://technet.microsoft.com/pt-br/library/cc737478(WS.10).aspx>. Acesso em: 09 ago. 2011.

O’FLYNN, C. **Downloading, installing and configuring winavr**. Disponível em: <http://winavr.sourceforge.net/install_config_WinAVR.pdf>. Acesso em: 11 ago. 2011.

PAGANO, P.; CHITINIS M.; LIPARI, G. **Real-time applications in wireless sensor networks.** Retis Lab. Scuela Superiore Sant’Anna. Pisa, Italy. Dec. 2007.

PERKINS, C.; ROYER, B. E.; DAS,S. **Ad hoc on-demand distance vector (AODV) routing, RCF 3561**.Tech. Rep., Internet Engineering Task Force, 2003.

POTTIE, G. Wireless sensor networks. In: **Information Theory Workshop, Ireland,** p. 139-140, June 1998.

_____; KAISER, W. J. **Wireless integrated network sensors**. Communications of the ACM, New York USA, v. 43, n. 5, p. 51-58, 2000.

SEMPREBOM, T. et al. Escalonamento dinâmico de compartimentos de tempo garantido em redes ieee 802.15.4. In: **XVIII - Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, Gramado, RS, maio 2010.

SIMON, G. et al. Sensor network-based countersniper system. In: **SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems**, New York, NY, USA, p. 1–12, ACM, 2004.

SRIVASTAVA; EUSTACE, **Atom: a system for building customized program analysis tools**. Digital Western Research Laboratory. Palo Alto, CA. 1994.

STANKOVIC, J. A. **Wireless sensor networks**. Computer Magazine. IEEE Computer Society, Los Alamitos, CA, USA. v. 41, n. 10, p. 92-95, 2008a.

_____. **When sensor and actuator networks cover the world**. ETRI, Daejeon, Korea, v. 30, n. 5, p. 627-633, Oct. 2008b.

SUNDANI, H. et al. Wireless **sensor network simulators – a survey and comparisons**. International Journal of Computer Networks (IJCN). Kuala Lampur, Malaysia. v. 2, n.5, p. 249-265, 2011.

TANENBAUM, A. S. **Organização estruturada de computadores**. 4. ed. Rio de Janeiro, LTC, 2001.

_____. **Redes de computadores**. 4. ed. Rio de Janeiro, Elsevier, 2003.

TOLLE, G. et al. A macroscope in the redwoods. In: **Proceedings of the 3rd international conference on Embedded networked sensor systems, SenSys '05**, New York, NY, USA, p. 51–63, ACM, 2005.

TUBAISHAT, M.; MADRIA, S. **Sensor networks: an overview**. IEEE Potentials Magazine, New York, NY, USA. v. 22, p. 20-23, April-May 2003.

VIEIRA, M. et al. Survey on wireless sensor network devices. In: **Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03.** IEEE Conference, Lisbon, Portugal. v. 1, p. 537–544, Sept. 2003.

VOLOSENCU, C.; MALITA V. Application of virtual instrumentation for sensor network monitoring. In: **Proceedings of the 2nd International Conference on Manufacturing Engineering, Quality and Production Systems (MEQAPS'10)**. Romania, P. 264-268, sep. 2010.

WILLIG, A. **Recent and emerging topics in wireless industrial communications: a selection**. IEEE Transactions on industrial informatics, New York, NY, USA, v. 4, p. 102–124, May 2008.

WOOD, P. **A survey of performance analysis tools**. Washington University in St. Louis. Report, 2008. Disponível em: < http://www.cs.wustl.edu/~jain/cse567-06/ftp/perf_tools.pdf >. Acesso em: 09 ago. 2011.

XING, K. et al. Safety warning based on highway sensor networks. In: **Wireless Communications and Networking Conference**. New Orleans, LA, USA.IEEE, v. 4, p. 2355 – 2361, Mar. 2005.

YANG, B. Study on security of wireless sensor network based on Zigbee standard. In: **Proceedings of the 2009 International Conference on Computational Intelligence and Security - v. 02, CIS '09.** Washington, DC, USA, p. 426–430, IEEE Computer Society, 2009.

YAO, Y.; GEHRKE, J. **The cougar approach to in-network query processing in sensor networks**. ACM SIGMOD Records. New York, NY, USA, v. 31, p. 9–18, Sept. 2002.

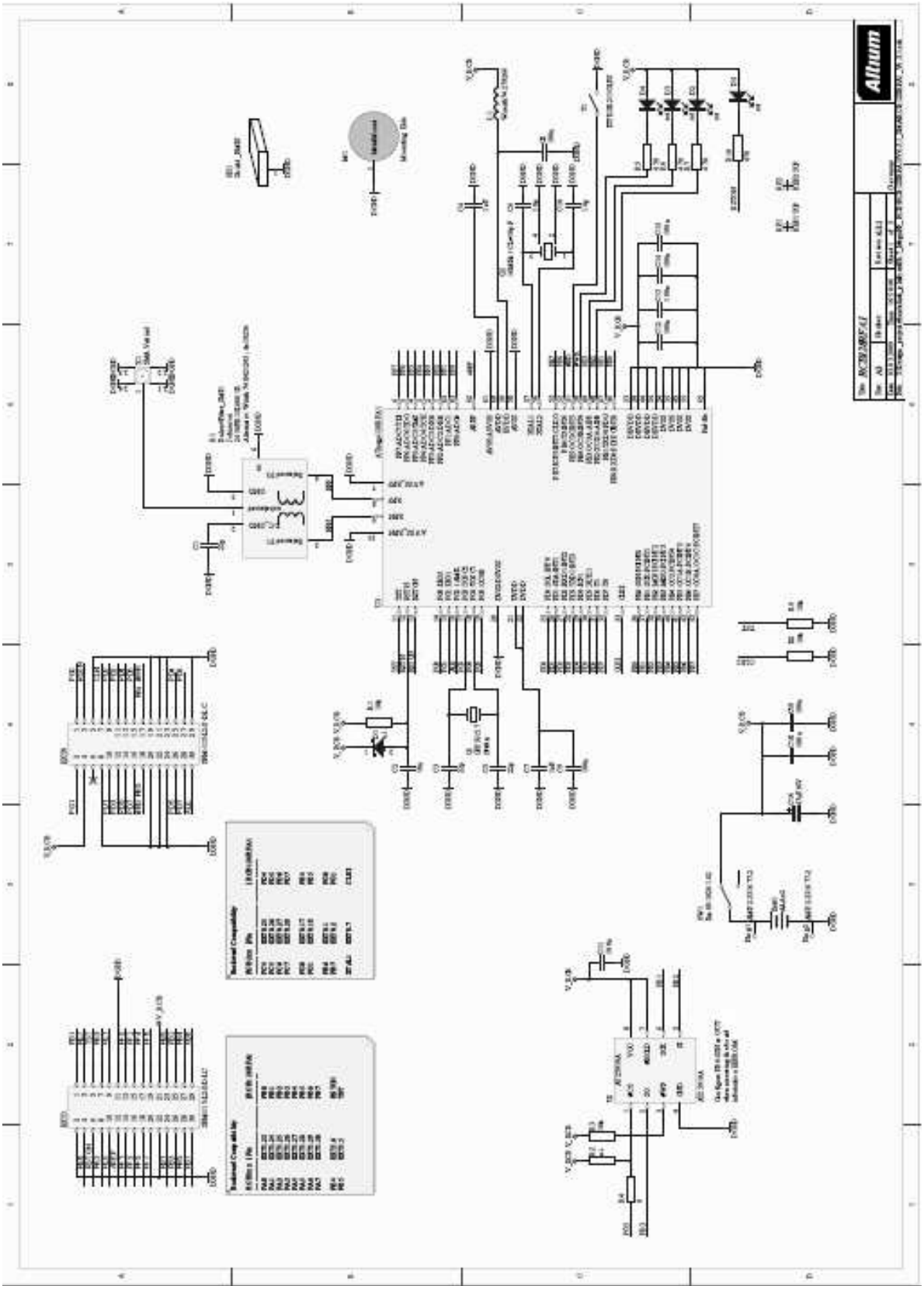
YEH, L-W.et al. **Autonomous light control by wireless sensor and actuator networks**. IEEE Sensors Journal, New York, NY, USA, v. 10,p. 1029–1041, June 2010.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. **Wireless sensor network survey**. ACM Computer Networks. New York, NY, USA, v. 52, p. 2292–2330, Aug. 2008.

ZIGBEE. **Zigbee specification**. Zigbee Alliance. San Ramon, CA, USA, Jan. 2008.

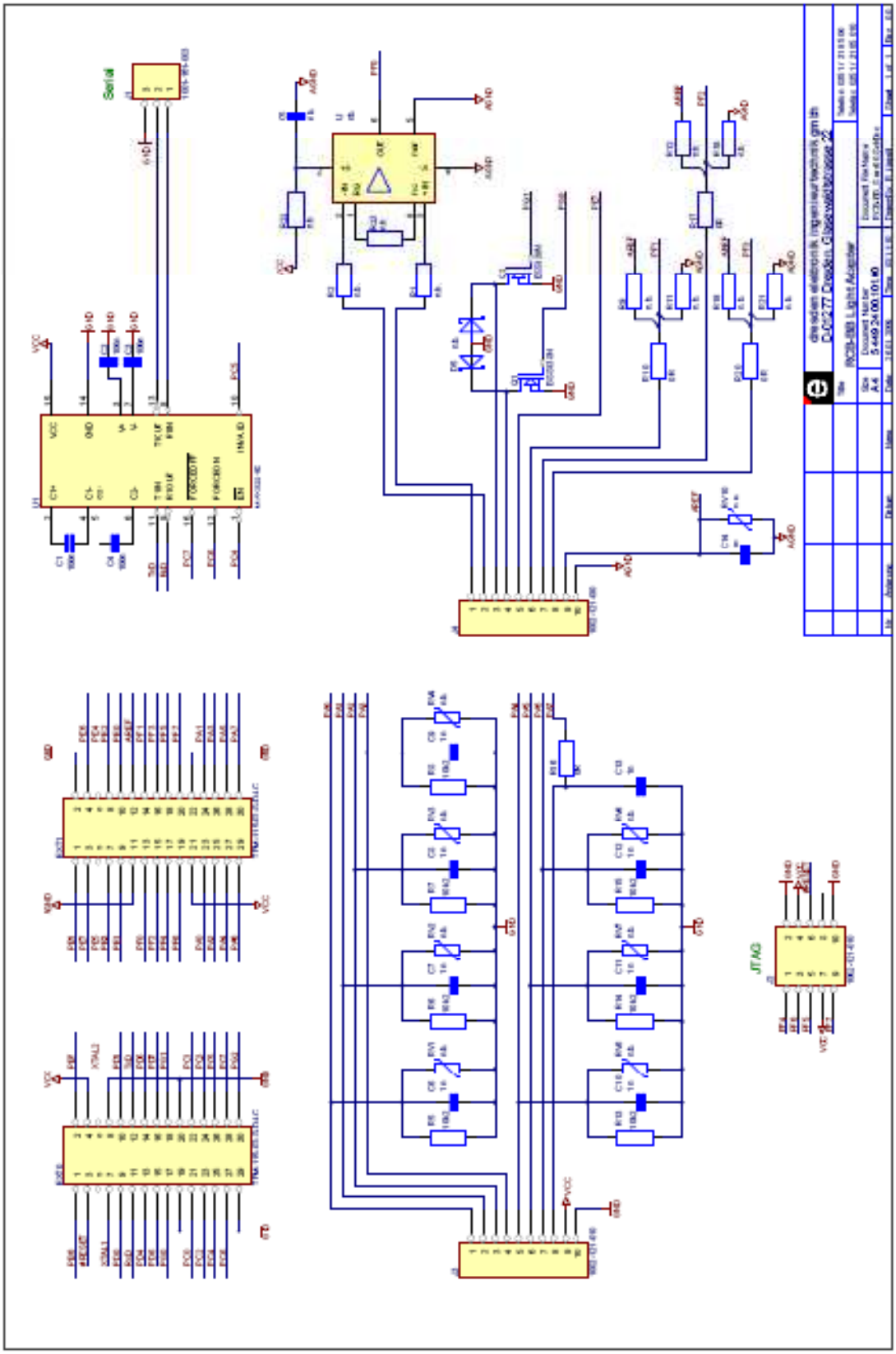
Anexo – A

Esquema elétrico da placa RCB128TFA1-V6.3.1



Anexo – B

Esquema elétrico da placa de interligação RCBBB



Anexo – C

Programa “sorvedouro.c” com as principais alterações introduzidas com o objetivo da realização da instrumentalização de software do nó sorvedouro. Com o objetivo de facilitar a localização das rotinas de instrumentalização do nó sorvedouro, as linhas 2, 3, 4, 5, 13, 14, 18 e 19 do pseudocódigo apresentado na figura 38 estão destacadas no programa fonte abaixo.

```
/**
 * @file main.c - Instrumentalização de Software do nó sorvedouro
 *
 * @brief  MAC Example - Basic Sensor Network
 *
 * This is the source code of a simple MAC example. It implements the
 * firmware for all nodes of a network with
 *
 *                        star topology
 *
 *      based on a Tiny MAC - _HIGHEST_STACK_LAYER = MAC
 *
 * with only minimal 802.15.4 primitive support.
 *
 * The actual supported primitives are defined in the corresponding file
 * "mac_user_build_config.h" in directory "Inc" of this application.
 *
 * In order to actually use this header file, the build switch
 * "MAC_USER_BUILD_CONFIG" needs to be set in the corresponding
 * Makefiles of this project.
 *
 * $Id: main.c 22913 2010-08-13 09:48:49Z sschneid $
 *
 * @author    Atmel Corporation: http://www.atmel.com
 * @author    Support email: avr@atmel.com
 */

/*
 * Copyright (c) 2009, Atmel Corporation All rights reserved.
 *
 * Licensed under Atmel's Limited License Agreement --> EULA.txt
 */

/* === INCLUDES
============================================================ */
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>
#include <ctype.h>
#include "pal.h"
#include "tal.h"
#include "tfa.h"
#include "app_config.h"
#include "ieee_const.h"
#include "bmm.h"
```

```c
#include "sio_handler.h"
#include "mac_api.h"

/* === TYPES
============================================================== */
typedef enum node_type_tag
{
    LEAF,
    ROOT_ROUTER,
    ROUTER,
    ROOT
} node_type_t;

typedef enum node_state_tag
{
    SEARCHING,
    CONNECTED,
    BROADCASTING
} node_state_t;

/* === MACROS
============================================================== */
#define TEMPERATURE_MEASUREMENT         (0)

/** Defines the default channel. */
#if (TAL_TYPE == AT86RF212)
    #ifdef CHINESE_BAND
        #define DEFAULT_CHANNEL         (0)
        #define DEFAULT_CHANNEL_PAGE    (5)
    #else
        #define DEFAULT_CHANNEL         (1)
        #define DEFAULT_CHANNEL_PAGE    (0)
    #endif  /* #ifdef CHINESE_BAND */
#else
#define DEFAULT_CHANNEL                 (20)
#endif  /* #if (TAL_TYPE == AT86RF212) */

/** Defines the PAN ID of the network. */
#define DEFAULT_PAN_ID                  (0xBEEF)

/** Defines the discovery duration during start-up */
#define DISCOVERY_DURATION_MS           (10000)

/** Defines the tx interval */
#define TX_INTERVAL_S                   (5)
//#define TX_INTERVAL_S                   (1)

/** Defines the receiver on duration */
#define RX_ON_DURATION_MS               (10)

/** Defines the delay (in us) between each broadcast message */
#define BROADCAST_DELAY                 (50000)

/** Define o tipo de teste a ser executado */
#define RTT_TEST                        (0)

//#define DEBBUGSINK                      (0)

#if (NO_OF_LEDS >= 3)
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_1)
```

```
#define LED_DATA_RX                     (LED_2)
#elif (NO_OF_LEDS == 2)
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_0)
#define LED_DATA_RX                     (LED_1)
#else
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_0)
#define LED_DATA_RX                     (LED_0)
#endif

/* === GLOBALS
============================================================== */
static node_type_t  node_type;
static uint64_t     parent_addr;
static node_state_t node_state;
static uint8_t      own_addr[8];
static uint16_t     NumSeqSrv      = 0;
static uint16_t     num_seq_pctenv = 0;
static uint16_t     NumPacEnv      = 0;
static uint16_t     IntEnvPac      = 0;
static uint8_t        CodOp          = 0x00;
static uint8_t        PrintForm      = 0x00;

/* === PROTOTYPES
============================================================= */
static void app_peripheral_init(void);
static bool button_pressed(void);
static void scan_timer_expired_cb(void *parameter);
#ifdef TEMPERATURE_MEASUREMENT
//static void adc_init(void);
#endif
//static uint16_t get_battery_value(void);
//static uint16_t get_temperature_value(void);
static void app_task(void);
static void send_broadcast(void);
static void send_roundtrip(void);

/* === IMPLEMENTATION
=================================================== */
/**
 * @brief Main function of the device application
 */
int main(void)
{
//-----------------------------------------------------------------------
// variável que contém o número seq. de pacote a ser enviados pela rede
    NumSeqSrv      = 0;
//-----------------------------------------------------------------------
// variável que contém o número seq. de pacotes enviados pelo Sorvedouro
    num_seq_pctenv = 0;
//-----------------------------------------------------------------------
// variável que contém o número de pacote a ser enviado pelo nó sensor
      NumPacEnv      = 2000;
//-----------------------------------------------------------------------
// variável que contém o intervalo de envio dos pacotes
    IntEnvPac      = 5000; // 1.000 milisegundos
//-----------------------------------------------------------------------
// variável que contém o código da operação
// = 0x01; RTT - Perfil dos aspectos temporais
// = 0x02  Perfil Manutenção da qualidade da Conexão
```

```
      CodOp          = 0x01; // RTT - Perfil dos aspectos temporais
//-----------------------------------------------------------------------
// variável que contém o tipo de impressão
// = 0x00 Print in ASCII form
// = 0x01 Print in Hex form
// = 0x02 Print in CSV form
    PrintForm      = 0x02;
//-----------------------------------------------------------------------
/* Initialize the MAC layer and its underlying layers, like PAL, TAL, BMM
*/
    if (wpan_init() != MAC_SUCCESS)
    {
        /*
         * Stay here; we need a valid IEEE address.
         * Check kit documentation how to create an IEEE address
         * and to store it into the EEPROM.
         */
        pal_alert();
    }

   /* Calibrate MCU's RC oscillator */
    pal_calibrate_rc_osc();

    /* Initialize LEDs */
    pal_led_init();
    pal_led(LED_ROOT, LED_OFF);     // indicating device is root
    pal_led(LED_DATA_TX, LED_OFF);  // indicating active period / data
transmit
    pal_led(LED_DATA_RX, LED_OFF);  // indicating data reception

    pal_led(LED_0, LED_ON);     // indicating application is started

    /*
     * The stack is initialized above, hence the global interrupts are
enabled
     * here.
     */

    pal_global_irq_enable();

//***********************************************************************

#ifdef SIO_HUB
/* Initialize serial interface used for communication with terminal program
*/
    if (pal_sio_init(SIO_CHANNEL) != MAC_SUCCESS)
    {
        // something went wrong during initialization
        pal_alert();
    }

#if ((!defined __ICCAVR__) && (!defined __ICCARM__) && (!(defined __GNUC__
&& defined __AVR32__))&& (!defined __GNUARM__))
    fdevopen(_sio_putchar, _sio_getchar);
#endif

    /* Setting stdin & stdout as unbuffered for AVR32 & ARM GCC*/
#if ((defined __GNUAVR32__) || (defined __GNUARM__))
    setbuf(stdin,NULL);
    setbuf(stdout,NULL);
#endif
```

```
#endif  /* SIO_HUB */

//*********************************************************************

    app_peripheral_init();
// linha 2 pseudocódigo sorvedouro.c "Configurar o hardware para operar como nó sorvedouro"

node_type = ROOT_ROUTER;
    wpan_mlme_reset_req(true);

   sio_getchar();

    /* Main loop */
    while (1)
    {
        wpan_task();
        app_task();
    }
}

/**
 * @brief Application task
 */
static void app_task(void)
{
    switch (node_state)
    {
        case BROADCASTING:
            break;

        case CONNECTED:
            if (node_type != LEAF)
            {
// linha 3 e 4 do pseudocódigo sorvedouro.c "Se o botão estiver pressionado".
              if(button_pressed())
                {
                node_state = BROADCASTING;
                send_broadcast();
                }
            }
            break;

        default:
            break;
    }

}

/**
 * @brief Callback function usr_mlme_reset_conf
 *
 * Função original wpan_mlme_reset_req
 *
 * @param status Result of the reset procedure
 */
void usr_mlme_reset_conf(uint8_t status)
{
    if (status == MAC_SUCCESS)
```

```
    {
        /* Get IEEE address */
        wpan_mlme_get_req(macIeeeAddress);
    }
    else
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
    }
}

/**
 * @brief Callback function usr_mlme_get_conf
 *
 * Função original wpan_mlme_get_req() / Primitiva para ler PIB atributos
 *
 * @param status            Result of requested PIB attribute get operation
 * @param PIBAttribute      PIB attribute
 * @param PIBAttributeValue PIB attribute value
 */
void usr_mlme_get_conf(uint8_t status,
                       uint8_t PIBAttribute,
                       void *PIBAttributeValue)
{
    if (status == MAC_SUCCESS)
    {
        if (PIBAttribute == macIeeeAddress)
        {
            /* Store own IEEE address to variable. */
            memcpy(own_addr, PIBAttributeValue, 8);

            /* Set default channel */
            uint8_t ch = DEFAULT_CHANNEL;
            wpan_mlme_set_req(phyCurrentChannel, &ch);
        }
    }
    else
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
    }
}

/**
 * @brief Callback function usr_mlme_rx_enable_conf
 *
 * @param status Result of the rx enable procedure
 */
void usr_mlme_rx_enable_conf(uint8_t status)
{
    /* Keep compiler happy. */
    status = status;
}

/**
 * @brief Callback function usr_mlme_set_conf
 *
 * Função original  wpan_mlme_set_req()
 * Primitiva paa escrever um determinado atributo no PIB
 *
 * @param status        Result of requested PIB attribute set operation
```

```
 * @param PIBAttribute  Updated PIB attribute
 */
void usr_mlme_set_conf(uint8_t status, uint8_t PIBAttribute)
{
    if (status != MAC_SUCCESS)
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
        return;
    }

    if (node_state != CONNECTED)
    {
        if (PIBAttribute == phyCurrentChannel)
        {
            /* Set default channel */
            uint16_t pan_id = DEFAULT_PAN_ID;
            wpan_mlme_set_req(macPANId, &pan_id);
        }
        else if (PIBAttribute == macPANId)
        {
            uint8_t pib_value;

            pib_value = true;

            node_state = SEARCHING;
            /* Switch receiver on to receive broadcast message */
            wpan_mlme_set_req(macRxOnWhenIdle, &pib_value);
            wpan_mlme_rx_enable_req(false, 0, 0xFFFFFF);
            pal_timer_start(APP_TIMER,
                            ((uint32_t)DISCOVERY_DURATION_MS * 1000),
                            TIMEOUT_RELATIVE,
                            (FUNC_PTR)scan_timer_expired_cb,
                            NULL);
        }
        return;
    }

}
/***************************************************************************
//
// Função de notificação de recebimento de mensagem
/**
 * @brief Callback function usr_mcps_data_ind
 *
 * Função original não tem - é um aviso de chegada de um pacote MPDU
 *
 * @param SrcAddrSpec      Pointer to source address specification
 * @param DstAddrSpec      Pointer to destination address specification
 * @param msduLength       Number of octets contained in MSDU
 * @param msdu             Pointer to MSDU
 * @param mpduLinkQuality  LQI measured during reception of the MPDU
 * @param DSN              DSN of the received data frame.
 * @param Timestamp        The time, in symbols, at which data were
received.
 *                         (only if timestamping is enabled).
 */
void usr_mcps_data_ind(wpan_addr_spec_t *SrcAddrSpec,
                       wpan_addr_spec_t *DstAddrSpec,
                       uint8_t msduLength,
                       uint8_t *msdu,
```

```
                        uint8_t mpduLinkQuality,
#ifdef ENABLE_TSTAMP
                        uint8_t DSN,
                        uint32_t Timestamp)
#else
                        uint8_t DSN)
#endif  /* ENABLE_TSTAMP */
{

    if (node_state == SEARCHING)
    {
        /* Check if this is an expected broadcast frame */
        if ((SrcAddrSpec->AddrMode == FCF_LONG_ADDR) &&
            (SrcAddrSpec->PANId    == DEFAULT_PAN_ID) &&
            (DstAddrSpec->AddrMode == FCF_SHORT_ADDR) &&
            (DstAddrSpec->PANId    == DEFAULT_PAN_ID))
        {
            /* Stop scan timer */
            pal_timer_stop(APP_TIMER);

            /* Store parent's address */
            ADDR_COPY_DST_SRC_64(parent_addr, SrcAddrSpec-
>Addr.long_address);
            node_state = CONNECTED;

/* Since a broadcast frame has been received, the device operated as
router. */
            node_type = ROUTER;
            /* The receiver is kept on. */
        }
    }
    else    /* Sensor data reception - node state is not SEARCHING */
    {
    pal_led(LED_DATA_RX, LED_TOGGLE);

/* ROOT */
#ifdef SIO_HUB
```

// linha 18 pseudocódigo sorvedouro.c "Formatar registro sorvedouro com dados de interesse".
// linha 19 pseudocódigo sorvedouro.c "Enviar registro sorvedouro com dados de interesse para o instrumento de software".

```
/* ROOT node prints received data to serial interface. */
   char EndNsr[10];
   char EndSrv[10];
   uint8_t  i;
   uint8_t  EDSrv;
   uint8_t  EDNsr;
   uint8_t  LqiSrv;
   uint8_t  LqiNsr;
   uint8_t  CodOpNsr;
   uint16_t TenBatSrv;
   uint16_t TenBatNsr;
   uint16_t NumSeqSrv;
   uint16_t NumSeqNsr;
   uint16_t TempSrv;
   uint16_t TempNsr;
   uint32_t RelTxSrv;
   uint32_t RelRxSrv;
   uint32_t RelRxNsr;
```

```
   uint32_t RelTxNsr;
   uint8_t  sdu[24];

//-----------------------------------------------------------------------
// Pegar o valor do Relógio recepção do sorvedouro (RelRxSrv)
      pal_get_current_time(&RelRxSrv);
//-----------------------------------------------------------------------
// Pegar o valor da tensão do sorvedouro (TenBatSrv)
      TenBatSrv = tfa_get_batmon_voltage();
//-----------------------------------------------------------------------
// Pegar o valor da Temperatura do sorvedouro (TempSrv)
      TempSrv = tfa_get_temperature();
//-----------------------------------------------------------------------
// Pegar o valor do LQI de recepção Sorvedouro (LqiSrv)
      LqiSrv = (uint8_t)mpduLinkQuality;
//-----------------------------------------------------------------------
// Pegar o valor da ED de recepção Sorvedouro (EDSrv)
      EDSrv = (uint8_t)PHY_ED_LEVEL;
//-----------------------------------------------------------------------
// Pegar a identificação da operação - (CodOPNsr)
      CodOpNsr = msdu[0];
//-----------------------------------------------------------------------
// Pegar o valor do endereço do nó sensor remoto - (EndNsr)
//-----------------------------------------------------------------------

// Pegar o valor do Relógio recepção do sorvedouro (RelTxSrv)
        RelTxSrv = (uint32_t)msdu[9] | ((uint32_t)msdu[10] << 8) |
((uint32_t)msdu[11] << 16) | ((uint32_t)msdu[12] << 24);
//-----------------------------------------------------------------------
// Pegar o valor do relógio no instante da recepção do nó sensor(RelRxNsr)
        RelRxNsr  = (uint32_t)msdu[13] | ((uint32_t)msdu[14] << 8) |
((uint32_t)msdu[15] << 16) | ((uint32_t)msdu[16] << 24);
//-----------------------------------------------------------------------
// Pegar o valor do relógio (RelTxNsr)
        RelTxNsr  = (uint32_t)msdu[17] | ((uint32_t)msdu[18] << 8) |
((uint32_t)msdu[19] << 16) | ((uint32_t)msdu[20] << 24);
//-----------------------------------------------------------------------
// Pegar o valor da tensão do nó sensor remoto - (TenBatNsr)
      TenBatNsr = (uint16_t)msdu[21] | (uint16_t)msdu[22] << 8;
//-----------------------------------------------------------------------
// Pegar o valor da Temperatura do nó sensor remoto - (TempNsr)
        TempNsr   = (uint16_t)msdu[23] | (uint16_t)msdu[24] << 8;
//-----------------------------------------------------------------------
// Pegar o valor do LQI recebido (LqiNsr)
      LqiNsr = (uint8_t)msdu[25];
//-----------------------------------------------------------------------
// Pegar o valor da ED recebida (EDNsr)
      EDNsr = (uint8_t)msdu[26];
//-----------------------------------------------------------------------
// Pegar o valor sequencial do pacote do nó sensor (NumSeqSrv)
      NumSeqSrv = (uint16_t)msdu[27] | (uint16_t)msdu[28] << 8;
//-----------------------------------------------------------------------
// Pegar o valor sequencial do pacote do sorvedouro (NumSeqNsr)
      NumSeqNsr = (uint16_t)msdu[29] | (uint16_t)msdu[30] << 8;
/* ----------------------------------------------------------------- */

if (PrintForm == 0x00) {
//-----------------------------------------------------------------------
//                              Print ASCII form
//-----------------------------------------------------------------------
// Registrar a identificação da operação - (CodOP)
```

```
      printf("\n: CodOp=%u ", CodOpNsr );
//-----------------------------------------------------------------------
//                    Registro dos dados do nó sensor
//-----------------------------------------------------------------------
// Registrar o valor do endereço do nó sensor remoto - (EndSrvNsr)
      printf(": EndNsr=");
      for (i = 0; i < 8; i++)
        {
        sprintf(EndNsr, "%.2X", msdu[8-i]);
        printf(EndNsr);
        }
//-----------------------------------------------------------------------
// Registrar o valor do relógio Sorvedouro (RelTxSrv)
      printf(": RelTxSrv=%ld ", RelTxSrv);
//-----------------------------------------------------------------------
// Registrar o valor do relógio no instante da recepção do nó
sensor(RelRxNsr)
      printf(": RelRxNsr=%ld ", RelRxNsr);
//-----------------------------------------------------------------------
// Registrar o valor do relógio (RelTxNsr)
      printf(": RelTxNsr=%ld ", RelTxNsr);
//-----------------------------------------------------------------------
// Registrar o valor da tensão do nó sensor remoto - (TenBatNsr)
      printf(": TenBatNsr=%.4u mV ", TenBatNsr);
//-----------------------------------------------------------------------
// Registrar o valor da Temperatura do nó sensor remoto - (TempNsr)
      printf(": TempNsr=%.4u °C ", TempNsr);
//-----------------------------------------------------------------------
// Registrar o valor do LQI recebido (LqiNsr)
        printf(": LqiNsr=%u ", LqiNsr);
//-----------------------------------------------------------------------
// Registrar o valor da ED recebida (EDNsr)
        printf(": EDNsr=%u ", EDNsr);
//-----------------------------------------------------------------------
// Registrar o valor sequencial do pacote do nó sensor (NumSeqNsr)
      printf(": NumSeqNsr=%u ", NumSeqNsr);

//-----------------------------------------------------------------------
//              Registro dos dados do nó sorvedouro
//-----------------------------------------------------------------------
// Pegar o valor endereço IEEE do Nó Sorvedouro (EndSrv)
      printf(": EndSrv=");
      for (i = 0; i < 8; i++)
        {
        sprintf(EndSrv, "%.2X", own_addr[7-i]);
        printf(EndSrv);
        }
 //-----------------------------------------------------------------------
// Registrar o valor do relógio Sorvedouro quando da transmissão TX
(RelTxSrv)
      printf(": RelTxSrv=%.10ld;", RelTxSrv);
//-----------------------------------------------------------------------
// Registrar o valor do Relógio recepção do sorvedouro (RelRxSrv)
      printf(": RelRxSrv=%ld ", RelRxSrv);
//-----------------------------------------------------------------------
// Registrar o valor da tensão do sorvedouro (TenBatSrv)
      printf(": TenBatSrv=%.4u mV ", TenBatSrv);
//-----------------------------------------------------------------------
// Registrar o valor da Temperatura do sorvedouro (TempSrv)
      printf(": TempSrv=%.4u °C ", TempSrv);
//-----------------------------------------------------------------------
```

```
// Pegar o valor do LQI de recepção Sorvedouro (LqiSrv)
      printf(": LqiSrv=%u ", LqiSrv);
//-----------------------------------------------------------------------
// Pegar o valor da ED de recepção Sorvedouro (EDSrv)
      printf(": EDSrv=%u ", EDSrv);
//-----------------------------------------------------------------------
// Pegar o valor sequencial do pacote do sorvedouro (NumSeqSrv)
       printf(": NumSeqSrv=%u ", NumSeqSrv);
//-----------------------------------------------------------------------
}  /* PRINTASCIIFORM */

if (PrintForm == 0x01) {
//-----------------------------------------------------------------------
//                              Print HEXADECIMAL form
//-----------------------------------------------------------------------
//                       Registro dos dados do nó sorvedouro
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
      sdu[0] = (uint8_t)(EndSrv[0]);
      sdu[1] = (uint8_t)(EndSrv[1]);
      sdu[2] = (uint8_t)(EndSrv[2]);
      sdu[3] = (uint8_t)(EndSrv[3]);
      sdu[4] = (uint8_t)(EndSrv[4]);
      sdu[5] = (uint8_t)(EndSrv[5]);
      sdu[6] = (uint8_t)(EndSrv[6]);
      sdu[7] = (uint8_t)(EndSrv[7]);
//-----------------------------------------------------------------------
// Pegar o valor do Relógio recepção do sorvedouro (RelTxSrv)
      sdu[8] = (uint8_t)(RelTxSrv);   // LSB first
      sdu[9] = (uint8_t)(RelTxSrv >> 8);
      sdu[10] = (uint8_t)(RelTxSrv >> 16);
      sdu[11] = (uint8_t)(RelTxSrv >> 24); // MSB last
//-----------------------------------------------------------------------
// Pegar o valor do Relógio recepção do sorvedouro (RelRxSrv)
      sdu[12] = (uint8_t)(RelRxSrv);   // LSB first
      sdu[13] = (uint8_t)(RelRxSrv >> 8);
      sdu[14] = (uint8_t)(RelRxSrv >> 16);
      sdu[15] = (uint8_t)(RelRxSrv >> 24); // MSB last
//-----------------------------------------------------------------------
// Pegar o valor da tensão do sorvedouro (TenBatSrv)
      sdu[16] = (uint8_t)TenBatSrv;   // LSB first
      sdu[17] = (uint8_t)(TenBatSrv >> 8);
//-----------------------------------------------------------------------
// Pegar o valor da Temperatura do sorvedouro (TempSrv)
      sdu[18] = (uint8_t)TempSrv;   // LSB first
      sdu[19] = (uint8_t)(TempSrv >> 8);
//-----------------------------------------------------------------------
// Pegar o valor do LQI de recepção Sorvedouro (LqiSrv)
      sdu[20] = (uint8_t)LqiSrv;
//-----------------------------------------------------------------------//
Pegar o valor da ED de recepção Sorvedouro (EDSrv)
      sdu[21] = (uint8_t)EDSrv;
//-----------------------------------------------------------------------
// Pegar o valor Número sequencial do Sorvedouro (NumSeqSrv)
      sdu[22] = (uint8_t)NumSeqSrv;
      sdu[23] = (uint8_t)(NumSeqSrv >> 8);

      printf(">Hexa: ");
    printf("%.2X ", CodOpNsr);
    for (i = 0; i < msduLength; i++)
      {
```

```
      printf("%.2X ", msdu[i]);
      }
    printf("- ");
    for (i = 0; i < 24; i++)
      {
      printf("%.2X ", sdu[i]);
        }
    printf("\n<");
}  /* PRINTHEXAFORM  */

if (PrintForm == 0x02) {
//-----------------------------------------------------------------------
//                              Print form CSV
//-----------------------------------------------------------------------
//                       Registro dos dados do nó sensor
//-----------------------------------------------------------------------
// Registrar a identificação da operação - (CodOP)
      printf("\n>;%.2u;", CodOpNsr);
//-----------------------------------------------------------------------
// Registrar o valor endereço IEEE do Nó sensor remoto (EndNsr)
      for (i = 0; i < 8; i++)
        {
        sprintf(EndNsr, "%.2X", msdu[8-i]);
        printf(EndNsr);
        }
      printf(";");
//-----------------------------------------------------------------------
// Registrar o valor do relógio nó sensor no momento da recepção (RelRxNsr)
      printf("%.10ld;", RelRxNsr);
//-----------------------------------------------------------------------
// Registrar o valor do relógio nó sensor no momento da transmissão
(RelTxNsr)
      printf("%.10ld;;", RelTxNsr);
//-----------------------------------------------------------------------
// Registrar o valor da tensão do nó sensor remoto (TenBatNsr)
      printf("%.4u;", TenBatNsr);
//-----------------------------------------------------------------------
// Registrar o valor da Temperatura do nó sensor remoto (TempNsr)
      printf("%.4u;", TempNsr);
//-----------------------------------------------------------------------
// Registrar o valor do LQI recebido (LqiNsr)
      printf("%.3u;", LqiNsr);
//-----------------------------------------------------------------------
// Registrar o valor da ED (EDNsr)
      printf("%.3u;", EDNsr);
//-----------------------------------------------------------------------
// Registrar o valor sequencial do pacote do nó sensor (NumSeqNsr)
      printf("%.5u;", NumSeqNsr);

//-----------------------------------------------------------------------
// Registro dos dados do nó sorvedouro
//-----------------------------------------------------------------------
// Registrar o valor endereço IEEE do Nó sorvedouro (EndSrv)
      for (i = 0; i < 8; i++)
        {
        sprintf(EndSrv, "%.2X", own_addr[7-i]);
        printf(EndSrv);
        }
      printf(";");
//-----------------------------------------------------------------------
// Registrar o valor do relógio Sorvedouro quando da transmissão TX
```

```
(RelTxSrv)
      printf("%.10ld;", RelTxSrv);
//-----------------------------------------------------------------------
// Registrar o valor do Relógio recepção do sorvedouro (RelRxSrv)
      printf("%.10ld;;", RelRxSrv);
//-----------------------------------------------------------------------
// Registrar o valor da tensão do sorvedouro (TenBatSrv)
      printf("%.4u;", TenBatSrv);
//-----------------------------------------------------------------------
// Registrar o valor da Temperatura do sorvedouro (TempSrv)
      printf("%.4u;", TempSrv);
//-----------------------------------------------------------------------
// Registrar o valor do LQI de recepção Sorvedouro (LqiSrv)
      printf("%.3u;", LqiSrv);
//-----------------------------------------------------------------------
// Registrar o valor da ED de recepção Sorvedouro (EDSrv)
      printf("%.3u;", EDSrv);
//-----------------------------------------------------------------------
// Registrar o valor sequencial do pacote do sorv. recebido (NumSeqSrv)
      printf("%.5u;<", NumSeqSrv);
//-----------------------------------------------------------------------
}
/* PRINTCSVFORM  */
```

```
//-----------------------------------------------------------------------
#endif  /* SIO_HUB */
    }
    /* Keep compiler happy. */
    SrcAddrSpec     = SrcAddrSpec;
    DstAddrSpec     = DstAddrSpec;
    msduLength      = msduLength;
    msdu            = msdu;
    mpduLinkQuality = mpduLinkQuality;
    DSN             = DSN;
#ifdef ENABLE_TSTAMP
    Timestamp = Timestamp;
#endif  /* ENABLE_TSTAMP */
}

/**
 * Callback function usr_mcps_data_conf
 *
 * Função Original = wpan_mcps_data_req()
 *
 * @param msduHandle  Handle of MSDU handed over to MAC earlier
 * @param status      Result for requested data transmission request
 * @param Timestamp   The time, in symbols, at which the data were
transmitted
 *                    (only if timestamping is enabled).
 *
 */
#ifdef ENABLE_TSTAMP
void usr_mcps_data_conf(uint8_t msduHandle, uint8_t status, uint32_t
Timestamp)
#else
void usr_mcps_data_conf(uint8_t msduHandle, uint8_t status)
#endif  /* ENABLE_TSTAMP */
{
    if (node_state == BROADCASTING)
    {
        node_state = CONNECTED;
```

```
        /* Stay here avoiding that broadcast messages are sent too
frequently. */
        pal_timer_delay(BROADCAST_DELAY);
        pal_timer_start(APP_TIMER,
                   (TX_INTERVAL_S * 1000000),
                   TIMEOUT_RELATIVE,
                   (FUNC_PTR)send_roundtrip,
                   NULL);
        return;
    }
// Se o CodOp for 0x01 (Round-Trip Time - RTT)
    if ( CodOp == 0x01)
        {
      if (num_seq_pctenv < NumPacEnv)
         {
        pal_timer_start(APP_TIMER,
           (TX_INTERVAL_S * 1000000),
           TIMEOUT_RELATIVE,
           (FUNC_PTR)send_roundtrip,
           NULL);
         }
       return;
         }
     /* Keep compiler happy. */
    msduHandle = msduHandle;
    status = status;
#ifdef ENABLE_TSTAMP
    Timestamp = Timestamp;
#endif  /* ENABLE_TSTAMP */
}

/**
 * Callback function that is called when the scan timer expires
 *
 * @param parameter  Pointer to parameter - unused
 */
static void scan_timer_expired_cb(void *parameter)
{

    /* No parent has been found. */
    if (node_type == ROOT_ROUTER)
    {
        node_type = ROOT;
        node_state = CONNECTED;

        pal_led(LED_ROOT,     LED_ON);
        pal_led(LED_DATA_TX, LED_ON);
        pal_led(LED_DATA_RX, LED_OFF);
    }

    /* Keep compiler happy */
    parameter = parameter;
}

/**
 * Determine if button is pressed
 *
 * @return true if button is pressed, else false
 */
static bool button_pressed(void)
{
```

```
    if (pal_button_read(BUTTON_0) == BUTTON_PRESSED)
    {
        return true;
    }
    else
    {
        return false;
    }
}

/**
 * Get the battery/supply voltage value
 *
 * @return Supply voltage as measured with TRX' battery monitor
 */
/*static uint16_t get_battery_value(void)
{
    return tfa_get_batmon_voltage();
}*/

/**
 * Initialize peripheral used by application
 */
static void app_peripheral_init(void)
{
    pal_button_init();
#ifdef TEMPERATURE_MEASUREMENT
//    adc_init();
#endif
}

/**
 * Create and send broadcast message used for network setup procedure
 */
static void send_broadcast(void)
{
    uint8_t src_addr_mode;
    wpan_addr_spec_t dst_addr;
    uint8_t payload;

    src_addr_mode               = WPAN_ADDRMODE_LONG;
    dst_addr.AddrMode           = WPAN_ADDRMODE_SHORT;
    dst_addr.PANId              = DEFAULT_PAN_ID;
    dst_addr.Addr.short_address = BROADCAST;

    wpan_mcps_data_req(src_addr_mode,
                       &dst_addr,
                       1 /* payload length */,
                       &payload,
                       0 /* msduHandle */,
                       WPAN_TXOPT_ACK);
    /*
     * This request will cause a mcps data confirm message ->
     * usr_mcps_data_conf
     */
}
```

```
// linha 13 pseudocódigo sorvedouro.c "Formatar pacote sonda sorvedouro".

/**
 * Create and send round trip message used for leaf return it
```

```
 * Preenche e envia o pacote sonda sorvedouro com dados para
 * a instrumentalização do perfil dos aspectos temporais RTT
 * CodOp = 0x01
 */
static void send_roundtrip(void)
{
    uint8_t src_addr_mode;
    wpan_addr_spec_t dst_addr;
//payload length
    uint8_t payload[19];
    uint32_t RelTxSrv;
    num_seq_pctenv += 1;

//-----------------------------------------------------------------------
// Quadro de dados sonda sorvedouro - 19 Bytes
// CodOp = 0x01 Aspetos Temporais
// CodOp = 0x02 Manutenção da qualidade da conexão
    payload[0] = CodOp;
//-----------------------------------------------------------------------
// EndSrv = Endereço do nó sorvedouro
    payload[1]  = (uint8_t)(own_addr[0]);
    payload[2]  = (uint8_t)(own_addr[1]);
    payload[3]  = (uint8_t)(own_addr[2]);
    payload[4]  = (uint8_t)(own_addr[3]);
    payload[5]  = (uint8_t)(own_addr[4]);
    payload[6]  = (uint8_t)(own_addr[5]);
    payload[7]  = (uint8_t)(own_addr[6]);
    payload[8]  = (uint8_t)(own_addr[7]);
//-----------------------------------------------------------------------
// RelTxSrv = Relógio do nó sorvedouro no instante da transmissão
    pal_get_current_time(&RelTxSrv);
    payload[9]  = (uint8_t)(RelTxSrv);       // LSB first
    payload[10] = (uint8_t)(RelTxSrv >> 8);
    payload[11] = (uint8_t)(RelTxSrv >> 16);
    payload[12] = (uint8_t)(RelTxSrv >> 24); // MSB last
//-----------------------------------------------------------------------
//  Número de pacotes a ser enviado pelo nó sensor (NumPacEnv)
    payload[13] = (uint8_t)(NumPacEnv);
    payload[14] = (uint8_t)(NumPacEnv >> 8);
//-----------------------------------------------------------------------
//  Intervalo de envio entre pacotes (IntEnvPac)
    payload[15] = (uint8_t)(IntEnvPac);
    payload[16] = (uint8_t)(IntEnvPac >> 8);
//-----------------------------------------------------------------------
//  value sequential number (NumSeqSrv)
    payload[17] = (uint8_t)(num_seq_pctenv);
    payload[18] = (uint8_t)(num_seq_pctenv >> 8);
//-----------------------------------------------------------------------
//  Values of transmission
    src_addr_mode                = WPAN_ADDRMODE_LONG;
    dst_addr.AddrMode            = WPAN_ADDRMODE_SHORT;
    dst_addr.PANId               = DEFAULT_PAN_ID;
    dst_addr.Addr.short_address = BROADCAST;
//-----------------------------------------------------------------------
```

```
// linha 14 pseudocódigo sorvedouro.c "Transmitir pacote sonda sorvedouro
para a rede".
     wpan_mcps_data_req(src_addr_mode,
                        &dst_addr,
```

```
                        19 /* payload length */,
                        payload,
                        0 /* msduHandle */,
                        WPAN_TXOPT_ACK);
    /*
     * This request will cause a mcps data confirm message ->
     * usr_mcps_data_conf
     */
}
/* EOF */
```

Anexo – D

Programa “sensor.c” com as principais alterações introduzidas com o objetivo da realização da instrumentalização de software do nó sensor. Com o objetivo de facilitar a localização das rotinas de instrumentalização do nó sensor, as linhas 2, 5, 6, 8, 14, 15, 17 e 18 do pseudocódigo apresentado na figura 39 estão destacadas no programa fonte abaixo.

```
/***************************************************************************
 * @file main.c - Instrumentalização de Software do nó sensor
 *
 * @brief  MAC Example - Basic Sensor Network
 *
 * This is the source code of a simple MAC example. It implements the
 * firmware for all nodes of a network with
 *
 *                        star topology
 *
 *      based on a Tiny MAC - _HIGHEST_STACK_LAYER = MAC
 *
 * with only minimal 802.15.4 primitive support.
 *
 * The actual supported primtives are defined in the corresponding file
 * "mac_user_build_config.h" in directory "Inc" of this application.
 *
 *  In  order  to  actually  use  this  header  file,  the  build  switch
 * "MAC_USER_BUILD_CONFIG" needs to be set in the corresponding Makefiles
 of this project.
 *
 * $Id: main.c 22913 2010-08-13 09:48:49Z sschneid $
 *
 * @author    Atmel Corporation: http://www.atmel.com
 * @author    Support email: avr@atmel.com
 */
/*
 * Copyright (c) 2009, Atmel Corporation All rights reserved.
 *
 * Licensed under Atmel's Limited License Agreement --> EULA.txt
 ***************************************************************************/
/* === INCLUDES ========================================================== */
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
#include "pal.h"
#include "tal.h"
#include "tfa.h"
#include "mac_api.h"
#include "app_config.h"
#include "ieee_const.h"
#include "sio_handler.h"
```

```
/* === TYPES ========================================================== */
typedef enum node_type_tag
{
    LEAF,
    ROOT_ROUTER,
    ROUTER,
    ROOT
} node_type_t;

typedef enum node_state_tag
{
    SEARCHING,
    CONNECTED,
    BROADCASTING
} node_state_t;

/* === MACROS ========================================================= */
/** Defines the default channel. */
#if (TAL_TYPE == AT86RF212)
    #ifdef CHINESE_BAND
        #define DEFAULT_CHANNEL         (0)
        #define DEFAULT_CHANNEL_PAGE    (5)
    #else
        #define DEFAULT_CHANNEL         (1)
        #define DEFAULT_CHANNEL_PAGE    (0)
    #endif  /* #ifdef CHINESE_BAND */
#else
#define DEFAULT_CHANNEL                 (20)
#endif  /* #if (TAL_TYPE == AT86RF212) */

/** Defines the PAN ID of the network. */
#define DEFAULT_PAN_ID                  (0xBEEF)

/** Defines the discovery duration during start-up */
#define DISCOVERY_DURATION_MS           (10000)

/** Defines the tx interval */
#define TX_INTERVAL_S                   (5)

/** Defines the receiver on duration */
#define RX_ON_DURATION_MS               (10)

/** Defines the delay (in us) between each broadcast message */
#define BROADCAST_DELAY                 (50000)

#if (NO_OF_LEDS >= 3)
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_1)
#define LED_DATA_RX                     (LED_2)
#elif (NO_OF_LEDS == 2)
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_0)
#define LED_DATA_RX                     (LED_1)
#else
#define LED_ROOT                        (LED_0)
#define LED_DATA_TX                     (LED_0)
#define LED_DATA_RX                     (LED_0)
#endif

/* === GLOBALS ======================================================== */
static node_type_t  node_type;
```

```
static uint64_t     parent_addr;
static node_state_t node_state;
static uint8_t      own_addr[8];
static uint16_t     num_seq_pctenv = 0;
static uint32_t     RelTxSrv       = 0;
static uint16_t     NumPacEnv      = 0;
static uint16_t     IntEnvPac      = 5000;
static uint16_t     NumSeqSrv;
static uint32_t     RelRxNsr;
static uint8_t      CodOp;
static uint8_t      LqiNsr;
static uint8_t      EDNsr;
/* === PROTOTYPES ===================================================== */
static void app_peripheral_init(void);
static bool button_pressed(void);
static void scan_timer_expired_cb(void *parameter);
//static void data_tx_start_cb(void *parameter);
static void send_sensor_data(void);
#ifdef TEMPERATURE_MEASUREMENT
//static void adc_init(void);
#endif
static uint16_t get_battery_value(void);
//static uint16_t get_temperature_value(void);
static void app_task(void);

/* === IMPLEMENTATION ================================================= */
/**
 * @brief Main function of the device application
 */
int main(void)
{
    // variável que contém o número sequencial de pacotes enviados
    num_seq_pctenv  = 0;
    //  variável  que  contém  o  número  sequencial  de  pacotes  sonda  nó
sorvedouro
    NumSeqSrv       = 0;
    // variável que contém o número de pacotes a ser enviado
    NumPacEnv       = 0;
    // variável que contém o intervalo de tempo entre os envios dos pacotes
    IntEnvPac       = 5000;
   // variável que contém o código da operação (CodOp)
    CodOp           = 0;

/* Initialize the MAC layer and its underlying layers, like PAL, TAL, BMM
*/
    if (wpan_init() != MAC_SUCCESS)
    {
        /*
         * Stay here; we need a valid IEEE address.
         * Check kit documentation how to create an IEEE address
         * and to store it into the EEPROM.
         */
        pal_alert();
    }
    /* Initialize LEDs */
    pal_led_init();
    pal_led(LED_ROOT, LED_OFF);     // indicating device is root
    pal_led(LED_DATA_TX,  LED_OFF);   //  indicating  active  period  /  data
transmit
    pal_led(LED_DATA_RX, LED_OFF);  // indicating data reception
    /*
```

```
     * The stack is initialized above, hence the global interrupts are
enabled
     * here.
     */
    pal_global_irq_enable();

//*********************************************************************
#ifdef SIO_HUB
/* Initialize serial interface used for communication with terminal program
*/
    if (pal_sio_init(SIO_CHANNEL) != MAC_SUCCESS)
    {
        // something went wrong during initialization
        pal_alert();
    }
#if ((!defined __ICCAVR__) && (!defined __ICCARM__))
    fdevopen(_sio_putchar, _sio_getchar);
#endif
#endif  /* SIO_HUB */
//*********************************************************************
    app_peripheral_init();
// linha 2 pseudocódigo sensor.c “Configurar o hardware para operar como nó
sensor”
     node_type = LEAF;
    /*
    /*
     * Reset the MAC layer to the default values
     * This request will cause a mlme reset confirm ->
     * usr_mlme_reset_conf
     */
    wpan_mlme_reset_req(true);

    /* Main loop */
    while (1)
    {
        wpan_task();
        app_task();
//          printf(".");
    }
}
/**
 * @brief Application task
 */
static void app_task(void)
{

}

/**
 * @brief Callback function usr_mlme_reset_conf
 *
 * @param status Result of the reset procedure
 */
void usr_mlme_reset_conf(uint8_t status)
{
    if (status == MAC_SUCCESS)
      {
        /* Get IEEE address */
        wpan_mlme_get_req(macIeeeAddress);
      }
      else
```

```
        {
          // something went wrong; restart
          wpan_mlme_reset_req(true);
        }
}
/**
 * @brief Callback function usr_mlme_get_conf
 *
 * @param status            Result of requested PIB attribute get operation
 * @param PIBAttribute      PIB attribute
 * @param PIBAttributeValue PIB attribute value
 */
void usr_mlme_get_conf(uint8_t status,
                       uint8_t PIBAttribute,
                       void *PIBAttributeValue)
{
    if (status == MAC_SUCCESS)
    {
        if (PIBAttribute == macIeeeAddress)
        {
            /* Store own IEEE address to variable. */
            memcpy(own_addr, PIBAttributeValue, 8);

            /* Set default channel */
            uint8_t ch = DEFAULT_CHANNEL;
            wpan_mlme_set_req(phyCurrentChannel, &ch);
        }
    }
    else
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
    }
}
/**
 * @brief Callback function usr_mlme_rx_enable_conf
 *
 * @param status Result of the rx enable procedure
 */
void usr_mlme_rx_enable_conf(uint8_t status)
{
    /* Keep compiler happy. */
    status = status;
}
/**
 * @brief Callback function usr_mlme_set_conf
 *
 * @param status        Result of requested PIB attribute set operation
 * @param PIBAttribute  Updated PIB attribute
 */
void usr_mlme_set_conf(uint8_t status, uint8_t PIBAttribute)
{
    if (status != MAC_SUCCESS)
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
        return;
    }

    if (node_state != CONNECTED)
    {
```

```
        if (PIBAttribute == phyCurrentChannel)
        {
            /* Set default channel */
            uint16_t pan_id = DEFAULT_PAN_ID;
            wpan_mlme_set_req(macPANId, &pan_id);
        }
        else if (PIBAttribute == macPANId)
        {
            uint8_t pib_value;

            if (node_type == LEAF)
            {
                pib_value = false;
            }
            else
            {
                pib_value = true;
            }
            node_state = SEARCHING;
            /* Switch receiver on to receive broadcast message */

            wpan_mlme_set_req(macRxOnWhenIdle, &pib_value);
            wpan_mlme_rx_enable_req(false, 0, 0xFFFFFF);
            pal_timer_start(APP_TIMER,
                            ((uint32_t)DISCOVERY_DURATION_MS * 1000),
                            TIMEOUT_RELATIVE,
                            (FUNC_PTR)scan_timer_expired_cb,
                            NULL);
        }
        return;
    }
}
//*************************************************************************
//
// Função de notificação de recebimento de mensagem
/**
 * @brief Callback function usr_mcps_data_ind
 *
 * @param SrcAddrSpec      Pointer to source address specification
 * @param DstAddrSpec      Pointer to destination address specification
 * @param msduLength       Number of octets contained in MSDU
 * @param msdu             Pointer to MSDU
 * @param mpduLinkQuality  LQI measured during reception of the MPDU
 * @param DSN              DSN of the received data frame.
 * @param Timestamp            Time, in symbols, at which the data were
received.
 *                         (only if timestamping is enabled).
 */
void usr_mcps_data_ind(wpan_addr_spec_t *SrcAddrSpec,
                       wpan_addr_spec_t *DstAddrSpec,
                       uint8_t msduLength,
                       uint8_t *msdu,
                       uint8_t mpduLinkQuality,
#ifdef ENABLE_TSTAMP
                       uint8_t DSN,
                       uint32_t Timestamp)
#else
                       uint8_t DSN)
#endif  /* ENABLE_TSTAMP */
{
    if (node_state == SEARCHING)
```

```
    {
        /* Check if this is an expected broadcast frame */
        if ((SrcAddrSpec->AddrMode == FCF_LONG_ADDR) &&
            (SrcAddrSpec->PANId    == DEFAULT_PAN_ID) &&
            (DstAddrSpec->AddrMode == FCF_SHORT_ADDR) &&
            (DstAddrSpec->PANId    == DEFAULT_PAN_ID))
        {
            /* Stop scan timer */
            pal_timer_stop(APP_TIMER);

            /* Store parent's address */
            ADDR_COPY_DST_SRC_64(parent_addr,                    SrcAddrSpec-
>Addr.long_address);
            node_state = CONNECTED;

            if (node_type == LEAF)
            {
                uint8_t pib_value = true;
//                pal_led(LED_DATA_TX, LED_OFF);
                wpan_mlme_set_req(macRxOnWhenIdle, &pib_value);
            }
        }
    }
    else    /* Sensor data reception - node state is not SEARCHING */
```

// linha 5, 6 e 8 do pseudocódigo sensor.c "Se pacote sonda sorvedouro for recebido" – 01 Aspectos temporais 02 Manutenção qualidade da conexão

```
//-----------------------------------------------------------------------
//   Recebimento de pacote sonda do nó sorvedouro
//-----------------------------------------------------------------------
    {
    // Pegar o relogio do último frame recebido - considerado o tempo LRX
    pal_trx_read_timestamp(&RelRxNsr);
//-----------------------------------------------------------------------
// Pegar a identificação da operação - (CodOP)
      CodOp = msdu[0];
//-----------------------------------------------------------------------

// Pegar o valor do Relógio recepção do sorvedouro (RelTxSrv)
    RelTxSrv = (uint32_t)msdu[9] | ((uint32_t)msdu[10] << 8) | \
      ((uint32_t)msdu[11] << 16) | ((uint32_t)msdu[12] << 24);
//-----------------------------------------------------------------------

// Pegar o valor da quantidade de pacotes sondas nó sensor (NumPacEnv)
      NumPacEnv    = (uint16_t)(msdu[13]) | ((uint16_t)(msdu[14]) << 8);
//-----------------------------------------------------------------------

// Pegar o valor do Intervalo de tempo de pacotes sondas nó sensor
(IntPacEnv)
    IntEnvPac   = (uint16_t)(msdu[15]) | ((uint16_t)(msdu[16]) << 8);
//-----------------------------------------------------------------------

// Pegar o valor do número sequencial do pacote sonda sorvedouro
(NumSeqSrv)
      NumSeqSrv = (uint16_t)(msdu[17]) | ((uint16_t)(msdu[18]) << 8);
//-----------------------------------------------------------------------
// Pegar o valor do LQI de recepção Sorvedouro (LqiSrv)
    LqiNsr = (uint8_t)mpduLinkQuality;
//-----------------------------------------------------------------------
// Pegar o valor da ED de recepção Sorvedouro (EDSrv)
    EDNsr = (uint8_t)PHY_ED_LEVEL;
```

```
//    printf("\n%.5u;", NumSeqSrv);
//    printf("\n%.2u;", CodOp);

    if ( CodOp == 0x01 )
      {
        printf("\n%.2u;", CodOp);
      send_sensor_data();
      }
    if ( CodOp == 0x02 )
      {
      if ( num_seq_pctenv <= NumPacEnv )
        {
        pal_timer_start(APP_TIMER,
                    (TX_INTERVAL_S * 1000000),
                    TIMEOUT_RELATIVE,
                    (FUNC_PTR)send_sensor_data,
                    NULL);
        }
      }
```

```
    }
    /* Keep compiler happy. */
    SrcAddrSpec     = SrcAddrSpec;
    DstAddrSpec     = DstAddrSpec;
    msduLength      = msduLength;
    msdu            = msdu;
    mpduLinkQuality = mpduLinkQuality;
    DSN             = DSN;
#ifdef ENABLE_TSTAMP
    Timestamp = Timestamp;
#endif  /* ENABLE_TSTAMP */
} /* void usr_mcps_data_ind */
```

// linha 14 e 17 do pseudocódigo sensor.c “Formatar pacote sonda nó sensor”

```
//-----------------------------------------------------------------------
//   Preenchimento do quadro de dados sonda do nó sensor
//-----------------------------------------------------------------------
static void send_sensor_data(void)
{
    uint8_t          src_addr_mode;
    wpan_addr_spec_t dst_addr;
// Payload 31 bytes
    uint8_t          payload[31];
    uint16_t         TenBatNsr;
    uint16_t         TempNsr;
    uint8_t          i;
      uint32_t         currenttimelocal;

    num_seq_pctenv += 1;

/* Create payload - Carga útil do quadro de dados do nó sensor*/
//-----------------------------------------------------------------------
// CodOp
   payload[0] = CodOp;
//-----------------------------------------------------------------------
// Preenchimento do endereço do nó sensor padrão IEEE 32 bits (EndNsr)
    for (i = 1; i < 9; i++)
      {
      payload[i] = own_addr[i-1];
```

```
      }
//-----------------------------------------------------------------------
// Preenchimento do Relogio de Transmissão do Sorvedouro (RelTxSrv)
    if ( CodOp == 0x02 ) {
     RelTxSrv = 0x00000000; }
    payload[9]  = (uint8_t)RelTxSrv;   // LSB first
    payload[10] = (uint8_t)(RelTxSrv >> 8);
    payload[11] = (uint8_t)(RelTxSrv >> 16);
    payload[12] = (uint8_t)(RelTxSrv >> 24); // MSB last
//-----------------------------------------------------------------------
// Preenchimento do Relogio de Recepção do nó sensor (RelRxNsr)
    if (CodOp == 0x02) {
      RelRxNsr = 0x00000000;  }
      payload[13] = (uint8_t)(RelRxNsr);   // LSB first
    payload[14] = (uint8_t)(RelRxNsr >> 8);
    payload[15] = (uint8_t)(RelRxNsr >> 16);
    payload[16] = (uint8_t)(RelRxNsr >> 24); // MSB last
//-----------------------------------------------------------------------
// Preenchimento do valor da bateria do nó sensor (TenBatNsr)
    TenBatNsr    = get_battery_value();
    payload[21]  = (uint8_t)TenBatNsr;   // LSB first
    payload[22]  = (uint8_t)(TenBatNsr >> 8);
//-----------------------------------------------------------------------
// Preenchimento do valor da temperatura (TempNsr)
    TempNsr = tfa_get_temperature();
    payload[23]     = (uint8_t)TempNsr;   // LSB first
    payload[24]     = (uint8_t)(TempNsr >> 8);
//-----------------------------------------------------------------------
// Preenchimento do valor LQI da recepção x (LqiNsr)
// Não existe recepção de frame quando CodOp = 0x02, logo LQI = 0xff;
    if ( CodOp == 0x02 ) {
      payload[25] = (uint8_t)0xff;
      }
      else {
           payload[25] = (uint8_t)LqiNsr;
           }
//-----------------------------------------------------------------------
// Preenchimento do valor ED da recepção x (EDNsr)
// Não existe recepção de frame quando CodOp = 0x02, logo ED = 0xff;
    if ( CodOp == 0x02 ) {
      payload[26] = (uint8_t)0xff;
      }
      else {
           payload[26] = (uint8_t)EDNsr;
           }
//-----------------------------------------------------------------------
//Carregar o valor Sequencial do pacotes recebido (NumSeqSrv)
    if (CodOp == 0x02) {
    NumSeqSrv   = 0x0000; }
    payload[27] = (uint8_t)NumSeqSrv;
    payload[28] = (uint8_t)(NumSeqSrv >> 8);
//-----------------------------------------------------------------------
//Carregar o número sequencial do pacote enviado (NumSeqNsr)
    payload[29] = (uint8_t)num_seq_pctenv;
    payload[30] = (uint8_t)(num_seq_pctenv >> 8);
//-----------------------------------------------------------------------
// Copiar o valor do relógio corrente - considerado tempo de TX - LTX
    pal_get_current_time(&currenttimelocal);
    payload[17] = (uint8_t)currenttimelocal;   // LSB first
    payload[18] = (uint8_t)(currenttimelocal >> 8);
    payload[19] = (uint8_t)(currenttimelocal >> 16);
```

```
    payload[20] = (uint8_t)(currenttimelocal >> 24); // MSB last
//-----------------------------------------------------------------------
    src_addr_mode     = WPAN_ADDRMODE_LONG;
    dst_addr.AddrMode = WPAN_ADDRMODE_LONG;
    dst_addr.PANId    = DEFAULT_PAN_ID;
    ADDR_COPY_DST_SRC_64(dst_addr.Addr.long_address, parent_addr);
```

// linha 15 e 18 do pseudocódigo sensor.c "Transmitir pacote sonda nó sensor para o sorvedouro"

```
      wpan_mcps_data_req(src_addr_mode,
                         &dst_addr,
                         31 /* payload length */,
                         payload,
                         0 /* msduHandle */,
                         WPAN_TXOPT_OFF);
      printf("\nsent---> %.2u;", CodOp);
      /*
     * This request will cause a mcps data confirm message ->
     * usr_mcps_data_conf
     */
}
/**
 * Callback function usr_mcps_data_conf
 *
 * Função Original = wpan_mcps_data_req
 *
 * @param msduHandle  Handle of MSDU handed over to MAC earlier
 * @param status      Result for requested data transmission request
 * @param Timestamp     The time, in symbols, at which the data were
transmitted
 *                    (only if timestamping is enabled).
 *
 */
#ifdef ENABLE_TSTAMP
void usr_mcps_data_conf(uint8_t msduHandle, uint8_t status, uint32_t
Timestamp)
#else
void usr_mcps_data_conf(uint8_t msduHandle, uint8_t status)
#endif  /* ENABLE_TSTAMP */
{

    printf("\n\n%.2u", status);

    if (node_state == BROADCASTING)
      {
      node_state = CONNECTED;
/* Stay here avoiding that broadcast messages are sent too frequently. */
      pal_timer_delay(BROADCAST_DELAY);
      return;
      }
```

// linha 18 do pseudocódigo sensor.c "Transmitir pacote sonda nó sensor para o sorvedouro"

```
      if ( CodOp == 0x02 )
      {
      if ( num_seq_pctenv < NumPacEnv )
        {
        pal_timer_start(APP_TIMER,
                   (TX_INTERVAL_S * 1000000),
```

```
                    TIMEOUT_RELATIVE,
                    (FUNC_PTR)send_sensor_data,
                    NULL);
        }
      return;
      }
    /* Keep compiler happy. */
    msduHandle = msduHandle;
    status     = status;
#ifdef ENABLE_TSTAMP
    Timestamp  = Timestamp;
#endif  /* ENABLE_TSTAMP */
}
/**
 * Callback function that is called when the scan timer expires
 *
 * @param parameter  Pointer to parameter - unused
 */
static void scan_timer_expired_cb(void *parameter)
{
    uint8_t i;
    /* No parent has been found. */
    if (node_type == ROOT_ROUTER)
    {
        node_type = ROOT;
        node_state = CONNECTED;

        pal_led(LED_ROOT, LED_ON);
        pal_led(LED_DATA_TX, LED_ON);
        pal_led(LED_DATA_RX, LED_OFF);
    }
    else    /* LEAF */
    {
        /* No parent has been found. Indicated it using LEDs. */
        while (1)
        {
            pal_led(LED_ROOT, LED_ON);
            for (i = 0; i < 2; i++)
            {
                pal_timer_delay(50000);
            }
            pal_led(LED_ROOT, LED_OFF);
            for (i = 0; i < 2; i++)
            {
                pal_timer_delay(50000);
            }
            if (button_pressed())
            {
                /* Re-start. */

                pal_led(LED_ROOT, LED_OFF);
                pal_led(LED_DATA_TX, LED_OFF);
                pal_led(LED_DATA_RX, LED_OFF);

                wpan_mlme_reset_req(true);
                break;
            }
        }
    }
    /* Keep compiler happy */
    parameter = parameter;
```

```
}
/**
 * Determine if button is pressed
 *
 * @return true if button is pressed, else false
 */
static bool button_pressed(void)
{
    if (pal_button_read(BUTTON_0) == BUTTON_PRESSED)
    {
        return true;
    }
    else
    {
        return false;
    }
}
/**
 * Get the battery/supply voltage value
 *
 * @return Supply voltage as measured with TRX' battery monitor
 */
static uint16_t get_battery_value(void)
{
    return tfa_get_batmon_voltage();
}
/**
 * Initialize peripheral used by application
 */
static void app_peripheral_init(void)
{
    pal_button_init();
}
/* EOF */
```