PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CEATEC

FACULDADE DE ENGENHARIA ELÉTRICA

DANIEL ADORNO GOMES

ARQUITETURA PARA INTERCONEXÃO DE REDES DE SENSORES SEM FIO E A INTERNET ATRAVÉS DE WEB SERVICES E O PROTOCOLO HTTP

PUC-CAMPINAS

2015

DANIEL ADORNO GOMES

ARQUITETURA PARA INTERCONEXÃO DE REDES DE SENSORES SEM FIO E A INTERNET ATRAVÉS DE WEB SERVICES E O PROTOCOLO HTTP

Dissertação apresentada como exigência para obtenção do Título de Mestre em Engenharia Elétrica, ao Programa de Pós Graduação Stricto Sensu em Engenharia Elétrica, Pontifícia Universidade Católica de Campinas.

Orientador: Prof. Dr. David Bianchini

PUC-CAMPINAS

2015

Ficha Catalográfica
Elaborada pelo Sistema de Bibliotecas e
Informação - SBI - PUC-Campinas

t621.3851
G633a

Gomes, Daniel Adorno.
Arquitetura para interconexão de redes de sensores sem fio e a Internet através de Web Services e o protocolo HTTP / Daniel Adorno Gomes. - Campinas: PUC-Campinas, 2015.
113p.

Orientador: David Bianchini.
Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Pós-Graduação em Engenharia Elétrica.
Inclui anexo e bibliografia.

1. Redes de sensores sem fio. 2. World Wide Web (Sistemas de recuperação da informação. 3. Internet (Redes de computação). 4. Interconexão de redes (Telecomunicações). I. Bianchini, David. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologias. Pós-Graduação em Engenharia Elétrica. III. Título.

22.ed. CDD – t621.3851

DANIEL ADORNO GOMES

ARQUITETURA PARA INTERCONEXÃO DE REDES DE SENSORES SEM FIO E A INTERNET ATRAVÉS DE WEB SERVICES E O PROTOCOLO HTTP

Dissertação apresentada ao Curso de Mestrado Profissional em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias da Pontifícia Universidade Católica de Campinas como requisito parcial para obtenção do título de Mestre Profissional em Gestão de Redes de Telecomunicações.

Área de concentração: Gestão de Redes e Serviços

Orientadora: Prof. Dr. David Bianchini

Dissertação defendida e aprovada em ____ de _____ de_____ pela

Comissão Examinadora constituída dos seguintes professores:

_____

Profa. Dr. David Bianchini

Orientador da Dissertação e Presidente da Comissão Examinadora

Pontifícia Universidade Católica de Campinas

_____

Prof. Dr. Paulo Sérgio Martins Pedro

Universidade Estadual de Campinas

_____

Prof. Dr. Omar Carvalho Branquinho

Pontifícia Universidade Católica de Campinas

Dedico este trabalho à minha filha Malú Del Santo Adorno e à minha esposa Mércia Maria Del Santo.

AGRADECIMENTOS

Em primeiro lugar, agradeço à Deus,

Pela VIDA e a oportunidade de concluir este trabalho;

Ao Prof. Dr. David Bianchini,

Meu orientador, pelo apoio, atenção, conhecimento compartilhado, paciência e amizade;

Ao Prof. Dr. Omar Carvalho Branquinho,

Por todo o conhecimento e orientações que me foram passadas sobre as redes de sensores sem fio, e por inspirar este trabalho;

À todos os mestrandos e professores,

Pelo apoio, amizade e por todo o conhecimento adquirido durante o curso;

Aos amigos Ricardo Marques, Neto Paim e Daniel Braga,

Pelos trabalhos realizados em grupo;

Ao amigo Neto Paim,

Por não me deixar desistir dessa batalha;

Ao amigo Daniel Braga,

Por todos os ensinamentos sobre engenharia elétrica;

À Pontifícia Universidade Católica de Campinas,

Pela bolsa concedida, permitindo que eu pudesse participar do curso de Mestrado Profissional em Gestão de Redes de Telecomunicações.

Se subo aos céus, lá estás; se faço a minha cama no mais profundo abismo,
lá estás também;

Salmo 139:8

RESUMO

GOMES, Daniel Adorno. Arquitetura Para Interconexão de Redes de Sensores Sem Fio e a Internet Através de Web Services e o Protocolo HTTP. 2015. Dissertação (Mestrado em Engenharia Elétrica) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas Ambientais e de Tecnologias, Programa de Mestrado Profissional em Gestão de Redes de Telecomunicações, Campinas, 2015.

Este trabalho apresenta uma proposta de arquitetura para interconexão de redes de sensores sem fio e a Internet utilizando somente a camada de aplicação, sem que haja a necessidade de alteração da pilha de protocolos de ambas as redes. A arquitetura baseia-se no protocolo HTTP, propondo a utilização da tecnologia de web services para interconectar os dois padrões de rede. A partir da instalação de uma rede de sensores sem fio num ambiente fechado, foi realizada a sua interconexão com a Internet com base na arquitetura proposta, utilizando tecnologias de padrão aberto, o protocolo para web services REST e uma estrutura de computação em nuvem da empresa Amazon Web Services. Foram coletadas informações para que se pudesse avaliar a funcionalidade, a confiabilidade e a eficiência do sistema, de acordo com a norma da ABNT, NBR ISO/IEC 9126. Os resultados apontaram para a viabilidade de implementação dessa proposta, pois, demonstrou-se que é possível a interconexão entre esses padrões de rede utilizando a camada de aplicação, de forma confiável, eficiente e com a utilização de recursos de baixo custo. Este estudo pode embasar propostas para estender a acessibilidade das redes de sensores aos dispositivos móveis.

Palavras-Chave — Rede de Sensores Sem Fio, RSSF, HTTP, Radiuino, Web services, REST, SOAP, interconexão, TCP/IP, Internet, Publisher.

ABSTRACT

This paper presents a proposal for architecture for interconnection of wireless sensor networks and the Internet using only the application layer, without the need to change the protocol stack of both networks. The architecture is based on the HTTP protocol, proposing the use of web services technology to interconnect the two network standards. From the installation of a wireless sensor network in a closed environment, its interconnection was performed with the Internet based on the proposed architecture using open standard technologies, the protocol for REST web services and cloud computing structure of the company Amazon Web Services. Information was collected to evaluate the functionality, reliability and system efficiency, accordingly to the ABNT standard, ISO / IEC 9126. The results showed the feasibility of implementing this proposal because it was shown that the interconnection is possible between these network standards using the application layer, reliably, efficiently and with the use of low cost resources. This study might encourage proposals to extend the accessibility of sensor networks to mobile devices.

Key Words — Wireless Sensor Network, WSN, HTTP, Radiuino, Web services, REST, SOAP, interconnection, internetworking, TCP/IP, Internet, Publisher.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

6Lowpan	=	IPv6 over Low-power Personal Area Networks
A/D	=	Conversor Analógico Digital
ABNT	=	Associação Brasileira de Normas Técnicas
ARM	=	Advanced RISC Machine
BER	=	Bit Error Rate
CORBA	=	Common Request Broker Architecture
dBm	=	Decibel miliwatt
DCOM	=	Distributed Component Object Model
DPM	=	Dynamic Power Management
DTN	=	Delay Tolerant Network
DVS	=	Dynamic Voltage Scaling
EPROM	=	Erasable Programmable Read-Only Memory
HTTP	=	Hypertext Transfer Protocol
IaaS	=	Infrastructure as a Service
IEEE	=	Institute of Electrical and Electronics Engineers
MGL	=	Módulo Gerenciador Local
MHz	=	Megahertz
MIL	=	Módulo de Integração Lógica
MMR	=	Módulo de Monitoramento Remoto
mW	=	Miliwatt

MWS	=	Módulo Web Services
PaaS	=	Platform as a Service
RAM	=	Randon Access Memory
REST	=	Representational State Transfer
RF	=	Rádio Feqüência
RMI	=	Remote Method Invocation
ROM	=	Read-Only Memory
RSSF	=	Rede de Sensor Sem Fio
RSSI	=	Received Signal Strength Indicator
SaaS	=	Software as a Service
SMR	=	Sistema de Monitoramento Remoto
SOA	=	Service Oriented Architecture
SOAP	=	Simple Object Access Protocol
SQL	=	Structured Query Language
TCP/IP	=	Transfer Control Protocol / Internet Protocol
UC	=	Unidade Computacional
UCP	=	Unidade Central de Processamento
UDDI	=	Universal Description, Discovery and Integration
UI	=	Unidade de Inteligência
URI	=	Uniform Resource Identifier
USB	=	Universal Serial BUS
WSDL	=	Web Services Description Language
XML	=	Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

Assim como havia sido previsto no final dos anos 90 e início dos anos 2000, tanto por órgãos de imprensa quanto por instituições de pesquisa do mundo todo, as redes de sensores sem fio (RSSF) se apresentam como uma das tecnologias mais importantes deste século (CHONG; KUMAR, 2003), e ainda com um horizonte vasto a ser explorado.

A tecnologia das redes de sensores sem fio integra elementos de sensoriamento, comunicação em rede, processamento computacional, armazenamento de informações, automação e controle. Através dessa tecnologia podemos viabilizar funcionalidades tais como monitoramento, atuação e controle em ambientes específicos (SOHRABY; MINOLI; ZNATI, 2007).

A utilização das redes de sensores sem fio vem ganhando cada vez mais importância em diversas áreas como a industrial, a hospitalar, a ambiental, a agrícola e a militar. Em todas essas áreas as RSSFs possuem aplicações práticas e reais, das quais podemos destacar as seguintes (SOHRABY; MINOLI; ZNATI, 2007):

- Médico-Hospitalares – monitoramento de pacientes, aplicação de medicamentos;
- Militares – estabelecimento de comunicação durante operações de guerra, detecção de ataques nucleares, biológicos e químicos;
- Ambientais – monitoramento de variáveis ambientais para prevenção de desastres naturais como enchentes e incêndios, por exemplo;
- Industriais – monitoramento de máquinas, tubulações e equipamentos em locais de difícil acesso ou que ofereçam risco à vida humana.
- Agrícolas: monitoramento e controle de estufas de hidroponia, controle de pragas e monitoramento de rebanhos.

Deve-se salientar, no entanto, que o acesso aos dados gerados por uma RSSF limita-se ao próprio ambiente monitorado, sendo necessária a integração das RSSFs com outros padrões de redes, como a TCP/IP, em especial a Internet, para

estender essa acessibilidade, permitindo que tanto a gerência dos dados quanto a gerência da rede sejam efetuadas de maneira remota (KARL; WILLIG, 2005).

Diante de contextos com os quais a sociedade se depara atualmente, como o da Internet das Coisas, que tem a tecnologia das redes de sensores sem fio como base para a sua implementação (GUBBIA et al., 2013). E ainda, o da mobilidade e acessibilidade proporcionada pelos *smartphones*, torna-se cada vez mais evidente a necessidade de interconexão das RSSFs com a Internet, de forma efetiva e direta, permitindo que os dados sejam disponibilizados aos usuários, tão logo sejam lidos pelos sensores (RAJESH et al., 2010), independentemente de sua localização física.

1.1. Motivação

Conforme citado no tópico anterior, existe uma crescente necessidade de interconexão das RSSFs com a Internet, permitindo que o gerenciamento dos dados e da rede sejam executados de forma remota e também para que sejam desenvolvidas soluções adequadas à realidade atual, onde as pessoas estão o tempo todo conectadas à Internet, praticamente em todos os locais e utilizando os mais diversos tipos de dispositivos que vão desde microcomputadores e notebooks, passando por *smartphones* e *tablets*, até os aparelhos de televisão. Essa adequação a uma proposta condizente com a atualidade, que remete à computação ubíqua e pervasiva, passa pela adoção de recursos de computação em nuvem e de tecnologias como a dos *web services*.

Existem no entanto, diversas formas de interconexão de RSSFs com redes TCP/IP como o IPv6 *over Low-power Personal Area Networks* (6Lowpan), o *Overlay-based*, o *application-level gateway*, o *Delay Tolerant Network* (DTN), o Virtual IP e o *Proxy*. Cada uma dessas formas possui implicações diferentes no que diz respeito a fatores como custo, escalabilidade, eficiência energética da RSSF e grau de dificuldade de implantação.

Baseado em tais fatores, este trabalho motiva-se pela definição de uma proposta de interconexão de RSSFs com a Internet que atue na camada de aplicação e que tenha como alicerce o protocolo HTTP, fazendo uso de recursos tecnológicos de computação em nuvem e *web services*.

1.2. Objetivo

Este trabalho tem como objetivo principal prover uma Arquitetura de interconexão entre Rede de Sensores sem fio - RSSF e a Internet que atue no nível da camada de aplicação com uso do Hypertext Transfer Protocol - HTTP, através da tecnologia de *web services*.

Para seu desenvolvimento, os objetivos secundários serão:

- Realizar uma implementação por meio de recursos de computação em nuvem;
- Que tenha baixo custo monetário de implementação com a utilização de padrões abertos como *open-hardware* e *open-source*.

O fato de ter como um de seus objetivos o baixo custo, no que diz respeito ao *hardware* e *software* empregados na sua implementação, reduz a gama de aplicações às quais a proposta apresentada por este trabalho poderá atender.

Tem-se como foco o atendimento a sistemas de monitoramento de menor porte, cujo as grandezas não necessitem ser monitoradas com uma freqüência elevada, como é o caso de estufas utilizadas para o cultivo de hortaliças (LI; YANG; WANG; GAO, 2011).

1.3. Organização do Trabalho

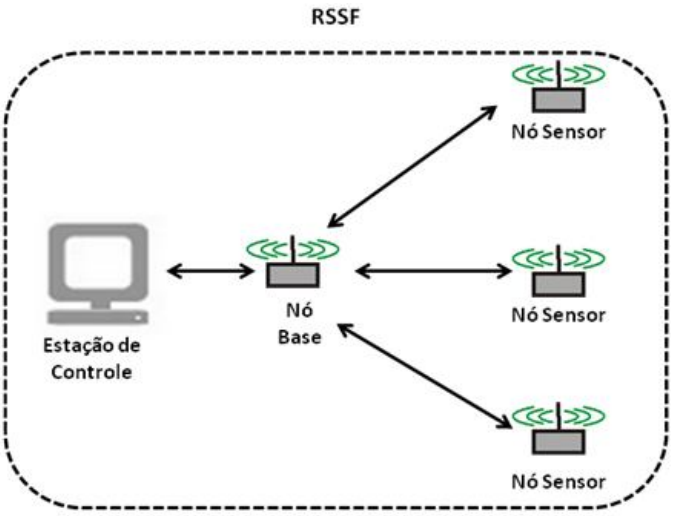
No capítulo 2, serão apresentados os conceitos da tecnologia das redes de sensores sem fio. No capítulo 3 serão apresentados alguns trabalhos relacionados ao tema. No capítulo 4, serão apresentadas as tecnologias de *web services* e de computação em nuvem. No capítulo 5, será apresentada a arquitetura proposta por este trabalho. No capítulo 6, é apresentada a metodologia adotada, os materiais utilizados e os resultados obtidos neste trabalho. No capítulo 7, estão as conclusões do trabalho.

2. A TECNOLOGIA DAS REDES DE SENSORES SEM FIO

Conforme citado na introdução deste trabalho, uma RSSF é uma infra-estrutura que compreende elementos de sensoriamento, comunicação em rede, processamento computacional e armazenamento de informações, fornecendo ao administrador funcionalidades tais como instrumentação, monitoramento, controle e reação a eventos que possam ocorrer em um determinado ambiente. Tipicamente, as aplicações das RSSFs estão relacionadas à coleta e ao acompanhamento de dados, atividades mais intimamente ligadas ao sensoriamento. No entanto, as RSSFs também podem ser implementadas de forma a prover recursos de atuação e controle (SOHRABY; MINOLI; ZNATI, 2007). As RSSFs podem ser projetadas para monitorar uma ou mais variáveis dentro de um ambiente específico, tais como temperatura, pressão, distância, direção, velocidade, umidade, vibração, luminosidade, som, atividade sísmica, velocidade do vento, peso, dentre outras (NAYAK; STOJMENOVIC, 2010).

Como ilustrado na Figura 1, a composição típica de uma RSSF envolve um determinado número de nós-sensores, os quais se encontram distribuídos dentro do espaço monitorado e associados por uma conexão sem fio a um nó-base, e uma estação de controle, normalmente representada por um microcomputador e um software capaz de gerenciar os dados enviados e recebidos da RSSF são os responsáveis pela leitura das variáveis no ambiente monitorado. Quando um nó-sensor obtém o valor de uma determinada variável, esse dado é enviado até à estação de controle através do nó-base. Ao receber a informação proveniente do nó-sensor, o software gerenciador deverá interpretá-la e, dependendo da situação, poderá executar algum tipo de ação, como diminuir a temperatura de uma estufa, por exemplo, ou simplesmente armazenar esse valor. Os nós-sensores podem ser fixos ou móveis, como por exemplo, aqueles utilizados em situações de guerra, conectados à soldados, veículos e robôs (SOUSA; LOPES, 2011) (CARVALHO et al, 2012).

Figura 1: Estrutura típica de uma RSSF.



Fonte: Elaboração própria.

2.1. Micro-controlador

O micro-controlador é a Unidade Central de Processamento (UCP) do nó-sensor. Basicamente, ele é responsável por todas as tarefas que um nó-sensor tem que executar como a coleta de dados, o processamento e o envio desses dados. A recepção de dados vindos de outros nós-sensores e o comportamento do atuador também é uma função do micro-controlador, além de executar diversos tipos de software, como por exemplo os protocolos de comunicação da camada de aplicação (KARL; WILLIG, 2005).

2.2. Memória

O componente de memória é formado basicamente pela memória de acesso aleatório ou memória RAM (*Randon Access Memory*), que armazena informações temporariamente como as leituras realizadas pelos nós-sensores e os pacotes de dados enviados e recebidos de outros nós. Existe também a necessidade de utilização um tipo de memória permanente como a ROM (*Read-Only Memory*) ou EPROM (*Erasable Programmable Read-Only Memory*), para que os softwares utilizados pelo nó-sensor fiquem armazenados mesmo depois de desligados. A memória do tipo flash também pode ser utilizada em substituição à EPROM, pois, além do armazenamento permanente, ela também pode ser utilizada para

armazenamento intermediário em caso de insuficiência ou ausência da RAM (KARL; WILLIG, 2005).

2.3. Sensores

Os sensores são dispositivos de *hardware* responsáveis pela captação de uma determinada grandeza, em resposta a uma alteração no estado físico, como por exemplo da temperatura ou luminosidade de uma determinada região monitorada. Eles produzem um sinal analógico de forma contínua, que é digitalizado por um conversor analógico-digital e enviado ao micro-controlador para processamento. Pode-se classificá-los em passivos omni-direcionais, passivos de feixe estreito e sensores ativos. Os sensores passivos omni-direcionais podem efetuar medições a partir de onde estão localizados sem manipular o ambiente, e não há direção envolvida nessas medições. Exemplos típicos de tais sensores incluem termômetro, sensores de luz, de vibração, microfones, umidade, dentre muitos outros. Já os sensores de feixe estreito, são passivos, mas tem uma noção bem definida da direção da medição. Os sensores ativos investigam o meio ambiente por meio de geração de eventos, como por exemplo, alguns tipos de sensores sísmicos, que produzem ondas de choque através de pequenas explosões (KARL; WILLIG, 2005).

2.4. Transceptor

Existem vários meios de comunicação pelos quais pode-se conectar os nós-sensores de uma RSSF, como rádio-freqüência, comunicações ópticas e ultra-som. Mas praticamente, o que tem sido utilizado é a comunicação baseada na freqüência de rádio, se adequando mais ao perfil das aplicações das RSSFs. Normalmente, as RSSFs utilizam freqüências de comunicação sem licença, que, em geral, são as seguintes: 868 e 915 MHz e 2,4 GHz. De forma objetiva, um nó-sensor necessita de um transmissor e um receptor para que a comunicação com outros nós-sensores possa ocorrer. Normalmente, essas duas funcionalidades são combinadas num único componente denominado transceptor, e cuja a tarefa é converter as cadeias de bits provenientes do micro-controlador em ondas de rádio, para que ocorra uma transmissão. Inversamente, para que ocorra uma recepção, é necessário que as ondas de rádio sejam convertidas em bits (KARL; WILLIG, 2005).

2.5. Fonte de energia

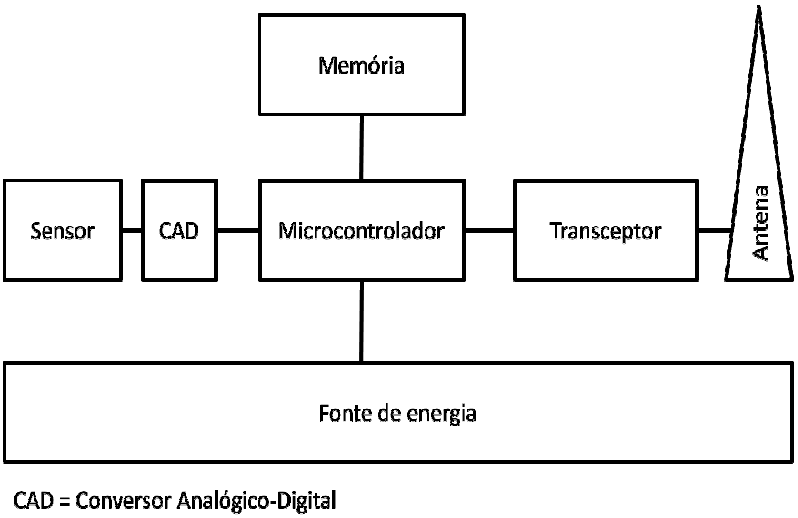
Um importante aspecto no desenvolvimento de uma RSSF é garantir que haja energia suficiente para alimentar os nós-sensores. O consumo de energia por um nó-sensor ocorre na detecção, comunicação e processamento de dados. A operação executada por um nó-sensor que consome mais energia é a transmissão de dados. Essa operação requer mais energia do que qualquer outro processo. Por exemplo, o custo da energia necessário para a transmissão de 1 Kb a uma distância de 100 metros, é aproximadamente o mesmo que o utilizado para a execução de 3 milhões de instruções (KARL; WILLIG, 2005).

Normalmente, a energia é armazenada em baterias ou capacitores, mas a opção de fornecimento de energia para nós-sensores mais utilizada são as baterias, tanto recarregáveis quanto não-recarregáveis. Como exemplo, podem ser citadas as baterias de níquel-cádmio, de níquel-zinco e de níquel-hidreto metálico. Existem experimentos que demonstram a utilização de fontes de energia alternativas como a solar, a gerada pelo vento e a energia proveniente da água utilizada num processo de irrigação (Morais et al, 2008).

Por se tratar de um componente crítico no funcionamento dos nós-sensores, o gerenciamento de energia é essencial. Duas políticas de economia de energia podem ser adotadas. O gerenciamento dinâmico de energia (*Dynamic Power Management* - DPM) e o gerenciamento dinâmico de escala de tensão (*Dynamic Voltage Scaling* - DVS). O DPM conserva a energia desligando partes do nó-sensor que não estão atualmente sendo utilizadas ou ativas. Já o DVS varia os níveis de potência dentro do nó-sensor, dependendo da carga de trabalho, e através dessa variação da tensão ao longo da freqüência tenta obter uma redução no consumo de energia (KARL; WILLIG, 2005).

Na Figura 2, pode-se observar um diagrama de blocos que representa as partes de um nó-sensor e como elas se integram.

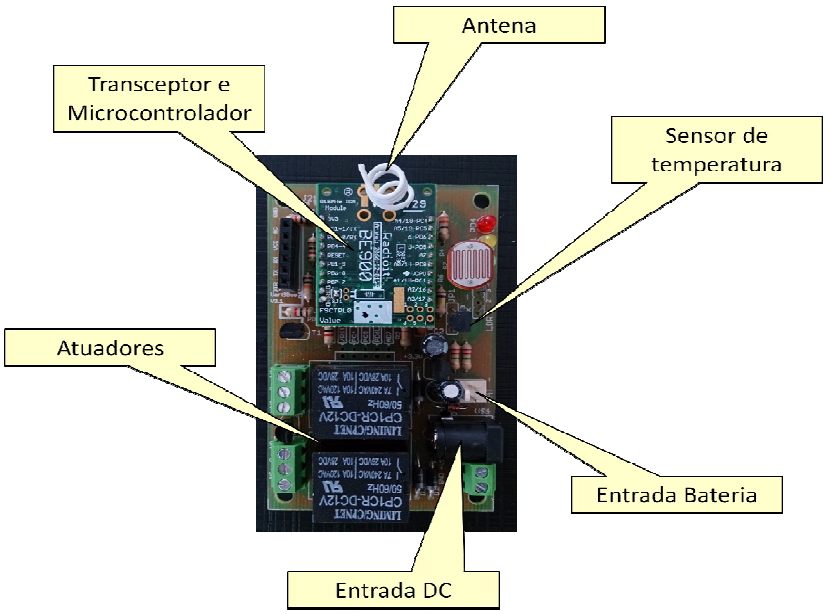
Figura 2: Arquitetura típica de um nó-sensor.



Fonte: (KARL; WILLIG, 2005).

A Figura 3, exibe uma foto de uma implementação real de um nó-sensor DK-103 produzido pela empresa Radio IT (RADIO IT, 2015).

Figura 3: Implementação real de um nó-sensor.



Fonte: Elaboração própria.

Quando os nós-sensores são idênticos entre si e executam as mesmas funções, a RSSF pode ser classificada como homogênea, mas no caso de RSSFs mais complexas compostas por tipos diferentes de nós-sensores, executando

funções diferentes, a rede é classificada como heterogênea (NAYAK; STOJMENOVIC, 2010).

A forma com que os nós-sensores e o nó-base trocam informações pode classificar uma RSSF em *single-hop* e *multi-hop*. Num cenário em que se utiliza poucos nós sensores num espaço de monitorado não tão grande, como uma estufa de plantas, por exemplo, normalmente se utiliza o *sigle-hop* ou de salto único, onde os dados coletados pelos sensores são transmitidos diretamente para o nó-base, que posteriormente, será enviada à estação de controle. Já num cenário com um número elevado de nós-sensores, onde a área de cobertura dos sensores é extensa e a distância entre os nós-sensores e o nó-base é muito grande, a opção que geralmente é utilizada para a comunicação é a *multi-hop* ou de múltiplos saltos. Nesse cenário, os nó-sensores mais distantes encaminham seus dados para outros nós-sensores mais próximos até que a informação chegue ao nó-base (NAYAK; STOJMENOVIC, 2010).

Com relação às aplicações que fazem uso de RSSFs, pode-se destacar as seguintes (NAYAK; STOJMENOVIC, 2010):

- Aplicações domésticas – ambientes inteligentes; automação de tarefas domésticas; segurança e monitoramento residencial;
- Aplicações médicas – monitoramento de pacientes e controle de medicamentos;
- Aplicações militares – estabelecimento de comunicação durante operações de guerra; detecção de ataques nucleares, biológicos e químicos;
- Aplicações ambientais – agricultura de precisão; monitoramento de desastres ambientais como enchentes, incêndios e pragas; monitoramento da qualidade da água;
- Aplicações industriais – monitoramento de máquinas, tubulações e equipamentos em refinarias ou em locais de difícil acesso.

3. A TECNOLOGIA DE *WEB SERVICES* E A COMPUTAÇÃO EM NUVEM

3.1. A Tecnologia de *Web Services*

Os *web services* são uma tecnologia de integração de sistemas projetada para suportar interoperabilidade do tipo *machine-to-machine* uma rede de computadores, mais especificamente a Internet (W3C WS, 2004).

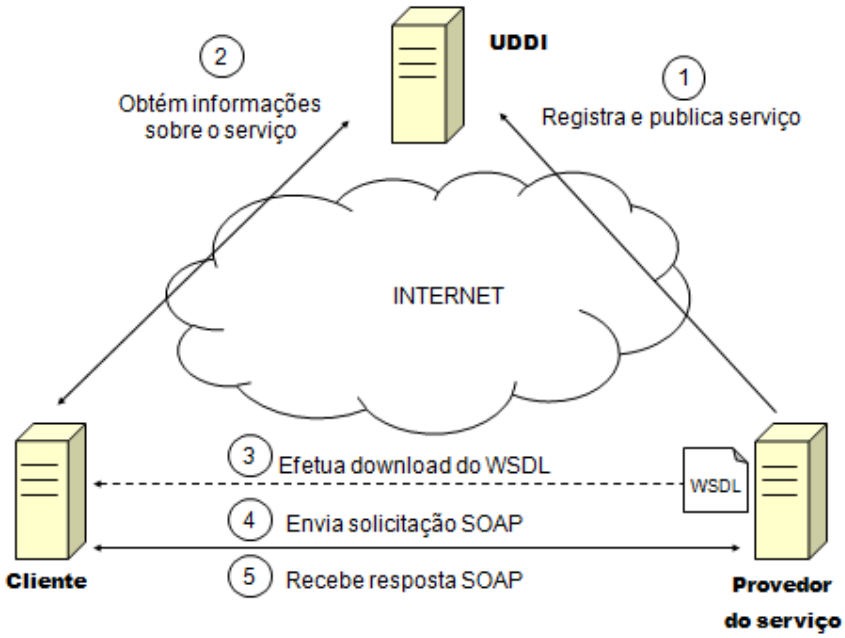
Essa tecnologia de padrão aberto, surgiu como uma evolução de tecnologias proprietárias de sistemas distribuídos, utilizadas durante a década de 90, como a *Common Request Broker Architecture* (CORBA), o *Remote Method Invocation* (RMI) e o *Distributed Component Object Model* (DCOM). A partir de um consórcio criado por algumas empresas pertencentes ao *World Wide Web Consortium* - W3C, essa tecnologia foi proposta com o objetivo de proporcionar uma meio padronizado de comunicação entre aplicações, através da Internet. Buscava-se um meio que permitisse a troca de informações entre aplicações independente de plataformas de hardware, software e de linguagem de programação. Foi dentro desse contexto que surgiu o primeiro protocolo para *web services* do mercado denominado *Simple Object Access Protocol* - SOAP, também conhecido como WS-* (HAMAD; SAAD; ABED, 2010).

O SOAP foi submetido ao W3C no ano de 2000 (W3C SUB, 2000). No mesmo ano, Roy Fielding, um dos autores do protocolo HTTP, apresentou em sua tese de doutorado, a proposta para um segundo protocolo voltado para *web services* denominado *Representational State Transfer* ou REST (FIELDING, 2000). Ambos os protocolos para web services utilizam o HTTP. A diferença é que o SOAP é executado sobre o protocolo HTTP, para trafegar suas mensagens no formato *Extensible Markup Language* - XML. Já o REST utiliza o próprio conteúdo do HTTP, seus métodos de envio e códigos de retorno, na troca de tais informações (RICHARDSON; RUBY, 2007).

3.1.1. *Web Services* SOAP

De acordo com a especificação do W3C, esses são os componentes envolvidos numa chamada à um *web service* SOAP, e a seqüência em que essa chamada acontece (W3C SOAP), conforme podemos observar na Figura 4:

Figura 4: Arquitetura dos web services SOAP especificada pelo W3C



Fonte: Modificado de (W3C SOAP).

SOAP: o *Simple Object Access Protocol*, é o protocolo padrão para transmissão de dados referente ao conjunto de especificações WS-*. Baseado no XML, o SOAP segue o modelo *REQUEST-RESPONSE* do HTTP.

WSDL: o *Web Services Description Language*, é um arquivo, padrão XML, que tem por finalidade fornecer uma descrição detalhada do *web service*, ao solicitante ou cliente. Essa descrição envolve a especificação das operações que compõem o serviço, definindo claramente como deve ser o formato de entrada e saída de cada operação. Como mostrado na figura, o WSDL pode tanto estar armazenado no Provedor de *web services*, quanto no UDDI.

UDDI: essa sigla significa Universal Description, Discovery and Integration. O UDDI é um mecanismo atende tanto ao cliente de *web services*, quanto ao

provedor. Ao provedor de *web services* o UDDI tem que fornecer recursos para que os *web services* sejam registrados e publicados, para que dessa forma, possam ser pesquisados e localizados pelos clientes de *web services*. O UDDI também pode ser utilizado para o armazenamento de arquivos WSDL.

Cliente: é um consumidor de *web services*, ou seja, um software que irá utilizar as operações de um determinado *web service*. Porém, na figura anterior é importante ressaltarmos que o cliente está representando várias estapas do ciclo de vida desse *software*. Desde sua pré-existência, quando o arquivo WSDL é obtido por um desenvolvedor, até o momento em que o *software* já está operando, onde ele faz solicitações e recebe os resultados dos *web services*.

Provedor de *web services*: é um application server ou um container web onde os web services ficam armazenados. Como pode-se verificar na Figura 4, ele pode armazenar também os arquivos WSDL.

A seguir, será descrita a sequência de operações realizadas por cada um dos componentes dessa arquitetura. Todas as informações trocadas entre qualquer uma das partes, são enviadas e recebidas através de mensagens no padrão XML:

Registra e publica o *web service*: o provedor de web services é o local onde os web services ficam armazenados, juntamente com os seus respectivos descritores, os arquivos WSDL. Quando um determinado *web service* é criado, ele é disponibilizado para utilização no provedor de *web services*. Porém, para que ele possa ser utilizado por algum cliente, ele e o seu WSDL precisam ser localizados, ou seja, o cliente precisa saber qual o endereço do serviço ou o seu URI (*Uniform Resource Identifier*). Dessa forma, após a criação e armazenamento de um *web service* no provedor, ele deve ser registrado e publicado num diretório de registro de *web services* ou UDDI.

Obtém informações sobre o *web service*: um cliente de *web services*, quando necessita utilizar um determinado serviço, primeiramente, irá pesquisar em diretórios de registro de *web services* (UDDI), pelo tipo de serviço desejado. Por exemplo, um *web service* que retorne a cotação do dólar. Os recursos de pesquisa e de localização de *web services* foram incluídos na arquitetura do W3C, pois, diversos *web services* de diversos fornecedores de *software*, podem dispobilizar a

mesma operação. Sendo assim, o cliente precisa obter a informação que dirá onde está o *web service* que ele deseja utilizar e o seu respectivo arquivo WSDL. Traduzindo, o UDDI irá fornecer o endereço (URI) do *web service* e do seu WSDL.

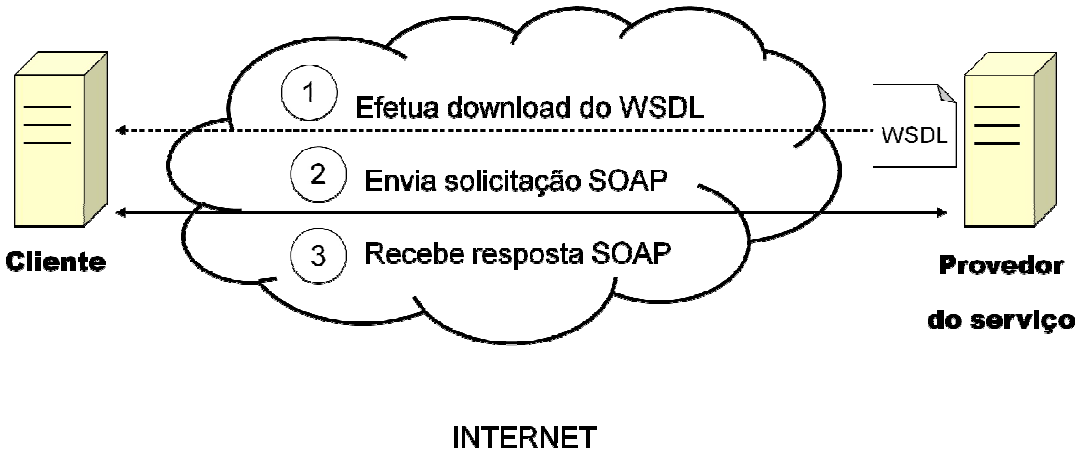
Efetua download do WSDL: após a obtenção dos URI's do *web service* e do seu descritor (WSDL), o cliente poderá efetuar o *download* do arquivo WSDL e prosseguir com a utilização do *web service* desejado. Não devemos esquecer que o arquivo WSDL pode estar disponível para *download* tanto no provedor de *web services*, quanto no UDDI. A partir da obtenção do arquivo WSDL e do URI do *web service*, um desenvolvedor terá condições de criar um software cliente que irá fazer uma chamada ao *web service* em questão e, em seguida, obter uma resposta.

Envia solicitação SOAP: com o desenvolvimento do *software* cliente, este irá enviar solicitações no padrão SOAP ao *web service*, referenciando o serviço através do seu URI.

Recebe resposta SOAP: após o envio da solicitação SOAP ao web service, normalmente, o software cliente irá receber uma resposta, também no padrão SOAP, como resultado da solicitação anterior.

A arquitetura descrita anteriormente, mostra o funcionamento do que foi especificado pelo W3C. Porém, é possível uma aplicação baseada no protocolo SOAP ser implementada sem a figura do UDDI, de forma que a comunicação se dá entre o cliente e o provedor de *web services*, sem intermediações, conforme pode-se observar na Figura 5.

Figura 5: Arquitetura dos web services SOAP sem UDDI

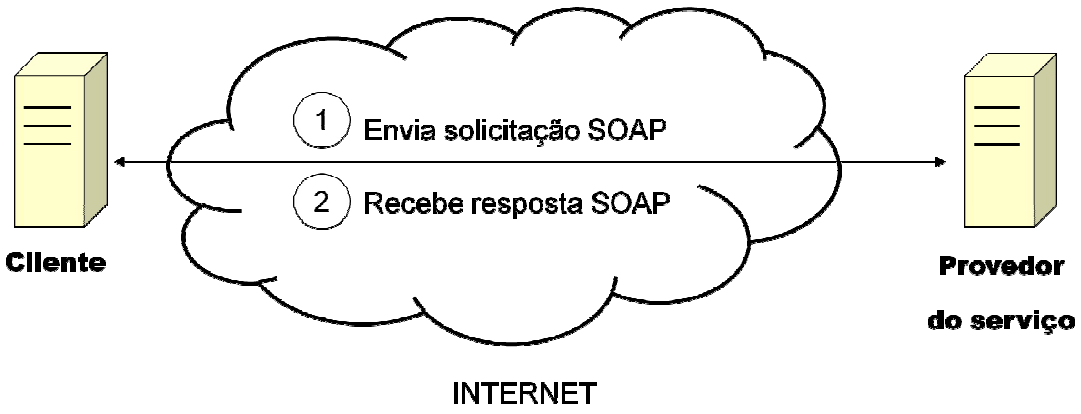


Fonte: Modificado de (W3C SOAP).

3.1.2. *Web Services* REST

A arquitetura de comunicação dos *web services* REST é mais simples do que a arquitetura proposta pelo W3C, conforme pode-se observar na Figura 6. Os componentes envolvidos são o Cliente de *web services*, o Provedor de *web services* e o protocolo HTTP (RICHARDSON; RUBY, 2007).

Figura 6: Arquitetura dos web services REST



Fonte: Modificado de (RICHARDSON; RUBY, 2007).

Diante desse cenário, tem-se o cliente enviando mensagens de solicitação à um determinado *web service* disponível no provedor. O *web service* em questão, irá realizar um determinado processamento e retornar uma mensagem de resposta ao cliente solicitante. O protocolo que determina o formato das mensagens enviadas e

recebidas, é o HTTP. No paradigma REST os serviços correspondem à recursos. Cada recurso corresponde à um URI que deve ser única e deve estar armazenada em um único local (RICHARDSON; RUBY, 2007).

Por exemplo, para um sistema de monitoramento de estufas, cada nó-sensor cadastrado corresponde a um recurso, ou seja, para cada um deles teremos um URI. O URI *http://www.monitoramento.com.br/nosensor/15* refere-se ao recurso nó-sensor cujo o código de identificação é 15. Essa estruturação provém, do fato dos *web services* REST serem totalmente baseados no HTTP, em seus métodos e em seus códigos de retorno.

Os métodos do HTTP correspondem às operações para a manipulação de dados relacionadas a cada recurso de um determinado sistema. E os códigos de retorno do HTTP, funcionam como um resultado da execução das operações de cada recurso (RICHARDSON; RUBY, 2007).

A Tabela 1 exibe os métodos do HTTP e a operação relacionada à cada um deles:

Tabela 1: Métodos do protocolo HTTP

Método	Descrição
GET	usado para obter um recurso ou uma lista deles
POST	usado para incluir um recurso
PUT	usado para editar um recurso
DELETE	usado para excluir um recurso

Fonte: (RICHARDSON; RUBY, 2007).

A tabela 2 exibe os principais códigos de retorno do HTTP utilizados em aplicações REST:

Tabela 2: Códigos de retorno do HTTP

Código	Significado
200	Sucesso
201	Criado
204	Sem conteúdo
400	Requisição inadequada
401	Não autorizado
403	Acesso negado
404	Não encontrado
412	Falha na pré-condição
500	Erro Interno no servidor
501	Não implementado

Fonte: (RICHARDSON; RUBY, 2007).

No momento em que se projeta um sistema baseado na arquitetura REST, primeiramente, defini-se os recursos envolvidos, posteriormente os métodos relacionados a cada um dos recursos, suas representações que farão parte do URI, e por fim, quais são os códigos HTTP retornados, que serão considerados para cada um dos métodos (RICHARDSON; RUBY, 2007).

3.1.3. Diferenças entre *web services* SOAP e REST

Os *web services* SOAP são utilizados, em sua maioria, com o protocolo HTTP POST, para o envio e recebimento de suas mensagens, as quais constituem arquivos no padrão XML. Nesse caso, o HTTP é o protocolo de aplicação sobre o qual as mensagens XML trafegam (W3C SOAP).

Já os *web services* REST, são totalmente baseados nos recursos oferecidos pelo HTTP, ou seja, o HTTP constitui ao mesmo tempo o protocolo de aplicação utilizado para trafegar a mensagem, bem como o seu recipiente (RICHARDSON; RUBY, 2007).

A principal discussão em torno desses dois padrões para construção de *web services*, é exatamente a grande quantidade de recursos consumida pelos *web*

services SOAP, por causa do peso do XML, em relação aos *web services* REST que, para a obtenção dos mesmos resultados, consomem uma quantidade de recursos muito menor, pois, o *parsing* XML nesse caso é bem menor ou mesmo nenhum, dependendo do *content-type* definido nas mensagens (POTTI; AHUJA; UMAPATHY, 2012).

3.2. A Computação em Nuvem

Pode-se definir a computação em nuvem como sendo um ambiente computacional no qual os aplicativos, a plataforma de desenvolvimento e a infra-estrutura são fornecidos no formato de serviços e acessados por meio da Internet, sob demanda e de forma flexível. Consiste em um sistema distribuído, composto por conjuntos de computadores virtualizados e interconectados. Esse ambiente apresenta-se como um ou mais recursos computacionais unificados, fornecidos dinamicamente ao consumidor (VOORSLUYS; BROBERG; BUYYA, 2011).

Baseado numa arquitetura orientada a serviços, os recursos dos ambientes de computação em nuvem são fornecidos aos consumidores via *Internet*, proporcionando flexibilidade e agilidade. É possível acrescentar ou retirar recursos rapidamente e o consumidor paga conforme o seu consumo (BUYYA; YEO; VENUGOPAL, 2008).

Os ambientes de computação em nuvem baseiam-se em tecnologias como *cluster* computacional, *grid computing*, virtualização, *Service Oriented Architecture* - SOA , *web services* e computação autonômica (VOORSLUYS; BROBERG; BUYYA, 2011).

Os serviços oferecidos num ambiente de computação em nuvem podem ser divididos em três categorias: infra-estrutura como serviço ou *Infrastructure as a Service* (IaaS), plataforma de desenvolvimento como Serviço ou *Platform as a Service* (PaaS) e aplicativos como serviço ou *Software as a Service* (SaaS) (TAURION, 2009).

Infrastructure as a Service (IaaS): são fornecidos recursos como servidores, armazenamento e comunicação na forma de serviços. A empresa Amazon, através da Amazon Web Services, oferece esse tipo de serviço (AWS, 2015)

Platform as a Service (PaaS): são fornecidos ambientes de desenvolvimento para que os clientes possam criar e hospedar suas próprias aplicações e distribuí-las como serviço. O IBM BlueMix é um exemplo desse tipo de plataforma. (IBM, 2015)

Software as a Service (SaaS): são fornecidos aplicativos aos usuários como se fossem serviços, acessados sob demanda, normalmente por meio de um navegador web. O Google Apps é um exemplo deste tipo de serviço (GOOGLE, 2015).

4. PROPOSTAS PARA A INTERCONEXÃO DE RSSFS COM A INTERNET

Sendo um ponto de grande importância, há na literatura diversas propostas de interconexão de RSSFs com redes TCP/IP, objetivando principalmente a extensão da acessibilidade à esse tipo de rede, que normalmente caracteriza-se por um acesso limitado ao ambiente que está sendo monitorado. Busca-se com isso, permitir que a gerência da rede e, no caso das RSSFs, dos dados monitorados, seja realizada de forma remota, mais especificamente via Internet.

Em (HO; FALL, 2004) os autores propõe a utilização do método *Delay Tolerant Network* (DTN) para prover interoperabilidade entre RSSFs e redes TCP/IP, como a Internet. Para proporcionar a comunicação entre redes heterogêneas através do DTN, uma camada denominada *bunddle layer* é adicionada acima da camada de transporte de ambas as redes interconectadas. O DTN utiliza um mecanismo do tipo *store-and-forward* para transmissão de pacotes, apresentando um bom desempenho da rede principalmente nos casos onde a perda de pacotes, os atrasos excessivos e a perda de conexão são comuns, como é o caso das RSSFs (KIM et al., 2007). Pode-se conectar RSSFs com a internet de maneira confiável com a utilização do DTN, mas é preciso levar consideração a alta complexidade para a implantação desse método (CHEN et al., 2009).

O *Overlay-based* é outro método de interconexão de redes heterogêneas. Esse método consiste na modificação da pilha de protocolos de uma das redes interconectadas, para que a comunicação entre elas seja possível. Em (DAI; HAN, 2004) é proposta a utilização da pilha de protocolos da RSSF sobre o TCP/IP. Dessa forma, cada *host* da rede TCP/IP passa a ser considerado um nó-sensor virtual, trocando informações diretamente com os nós-sensores da RSSF e processando pacotes como se fosse um deles. Uma característica desse método é o aumento da sobrecarga dos cabeçalhos do TCP/IP, os quais já são considerados pesados (SHU; WU; HU, 2006).

Ainda sobre o *Overlay-based*, em (DUNKELS et al., 2004) os autores propõem a implementação da pilha de protocolos TCP/IP em nós-sensores. No entanto, o TCP/IP mostra-se pouco adequado ao perfil das RSSFs. O fato do

cabeçalho TCP/IP ser muito pesado causa uma sobrecarga nos pacotes da RSSF, resultando num aumento do consumo de energia. Nessa mesma proposta, outro fator que leva a um aumento do consumo de energia por parte da RSSF, está relacionado a uma característica do TCP, que não apresenta uma boa performance em redes que onde a taxa de erro de bit (BER) é muito alta.

Conforme proposto em (LEI, S. et al), o método *Virtual-IP Gateway* não atribui um endereço IP físico diretamente aos nós-sensores de uma RSSF. O que ocorre é um mapeamento de um endereço IP lógico e o endereço de identificação de cada nó-sensor. Esse mapeamento fica armazenado no *gateway* que interconecta as duas redes permitindo a comunicação dos hosts da rede TCP/IP com nós-sensores específicos da RSSF. Segundo (DAMASO; DOMINGUES; ROSA, 2010), este método apresenta muitos problemas relacionados ao mapeamento do endereço lógico IP e de identificação dos nós-sensores.

A utilização do protocolo TCP/IP em RSSFs, de forma que cada nó-sensor possua um endereço IP e possa ser acessado diretamente, veio com a padronização do método de interconexão denominado *IPv6 over Low-power Personal Area Networks*, também conhecido como 6LowPAN, definido na RFC 4919 (USHALNAGAR; MONTENEGRO; SCHUMACHER, 2007) e adaptado para ser utilizado em RSSFs do padrão IEEE 802.15.4 (IEEE, 2011) na RFC 4944 (MONTENEGRO et al., 2007). Este método adiciona uma camada de adaptação abaixo da camada IP e permite que pacotes IPv6 sejam transmitidos sobre o protocolo utilizado na RSSF. Na camada de adaptação é criado um novo cabeçalho, a partir da compressão dos cabeçalhos das camadas IPv6 e de transporte, formado somente por alguns *bytes*. Dessa forma, torna-se viável a utilização do IPv6 em redes de sensores sem fio, pois, os pacotes são mais leves e o processamento exigido de cada nó-sensor é menor, se adequando mais ao perfil das RSSFs (HUI; CULLER, 2008). Apesar disso, questões relacionadas à eficiência energética continuam sendo um problema. A adaptação do datagrama IPv6 de 1.280 *Bytes* para 127 *Bytes* causa um excesso de fragmentação, o que implica em muito processamento por parte dos nós-sensores comprometendo a eficiência energética da RSSF (LUDOVICI; CALVERAS; CASADEMONT, 2011).

É essencial também, destacar que o 6LowPAN não conecta RSSFs a redes IPv6 diretamente, mas cria um mecanismo que permite aos nós-sensores da RSSF receber pacotes vindos de uma rede IPv6. O 6LowPAN não contempla o contrário, ou seja, os dispositivos de uma rede IPv6 não podem receber os pacotes provenientes de uma RSSF adaptada a este protocolo, pois não reconhecem a compressão, a fragmentação e os esquemas de endereçamento do 6LowPAN. Para que isso seja possível, é necessário, de forma complementar ao 6LowPAN, criar meios que permitam aos dispositivos de uma rede IPv6 receberem pacotes 6LowPAN enviados a partir de uma RSSF. Outra limitação apresentada pelo 6LowPAN, é a limitação imposta somente à dispositivos que utilizam o protocolo IPv6, excluindo aqueles utilizam o IPv4 (CAMPOS et al., 2011).

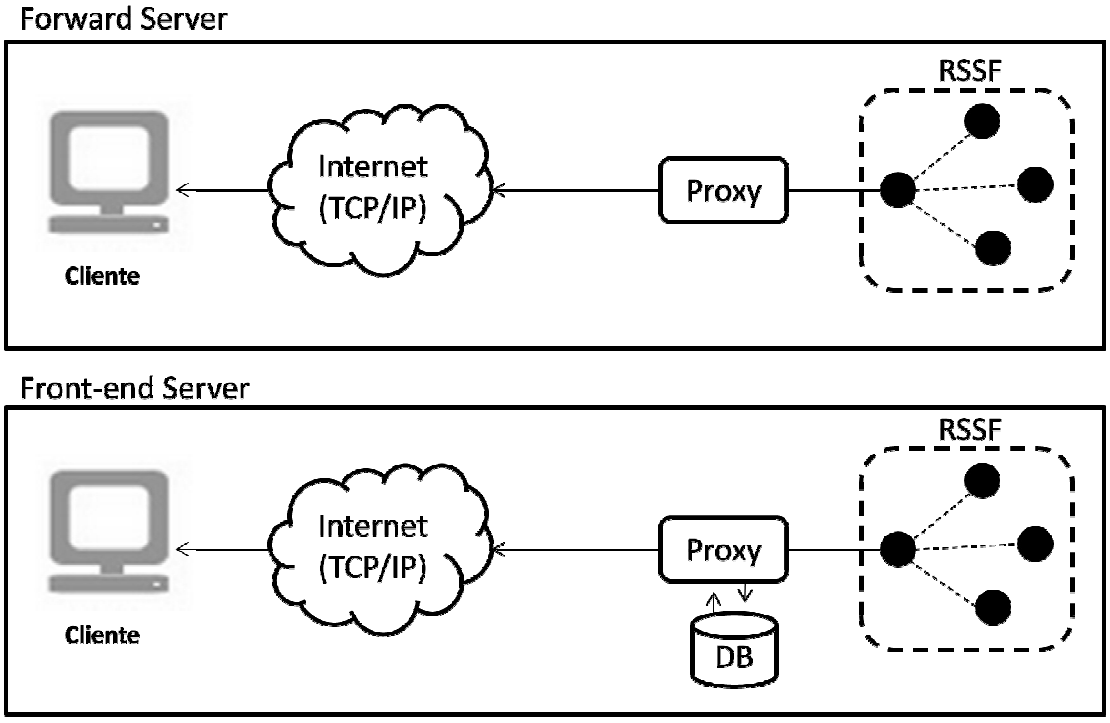
Dentre as opções de interconexão que atuam no nível da camada de aplicação, sem a necessidade de alteração da pilha de protocolos de nenhuma das redes estão o *gateway* (KARL; WILLIG, 2005) e o *proxy* (TING; XIAOYAN; YAN, 2009). Ambos os métodos isolam a rede de sensores da rede TCP/IP. Praticamente, um dispositivo físico, com capacidade de processamento computacional e armazenamento de informações, é instalado entre a RSSF e a rede TCP/IP, convertendo os dados vindos nos pacotes TCP/IP para o protocolo da RSSF e vice-versa, por meio de programação, permitindo que haja uma troca de informações de transparente entre as redes.

No caso do *gateway*, um *host* da rede TCP/IP solicita informações da RSSF. O gateway, por sua vez, solicita à RSSF a informação requisitada, utilizando o protocolo adequado. A RSSF retorna a informação em questão ao *gateway*, e este irá enviá-la ao host que iniciou a comunicação, novamente utilizando o protocolo TCP/IP.

Já o *proxy*, simplesmente envia as informações da RSSF para um ou mais *hosts* da rede TCP/IP, periodicamente, conforme pode-se observar na Figura 7. Dois padrões, normalmente, são empregados: o *forward server* e o *front-end server*. O primeiro método, somente transfere as informações da RSSF ao host da rede TCP/IP. Já o segundo, coleta os dados junto aos nós-sensores da RSSF e os armazena em um banco de dados, mas isso exige uma maior capacidade de armazenamento e processamento por parte do *proxy*. Nesse caso, o host da rede

TCP/IP pode obter os dados diretamente do banco de dados ou através de uma interface *web*, por exemplo. Isto reduz a latência e aumenta a vida útil da RSSF, uma vez que as informações são acessadas num banco de dados e não solicitadas diretamente aos nós-sensores (TING; XIAOYAN; YAN, 2009).

Figura 7: Métodos de interconexão baseados no conceito de proxy



Fonte: Elaboração própria.

Em geral, para ambos os métodos, um microcomputador é utilizado como o dispositivo físico que representa o *gateway* ou o *proxy*, mas existem propostas que fazem uso de *access points* como proposto em (CHEN et al., 2009) e em (MADEIRA, 2014). Este métodos apresentam outras vantagens como o baixo custo, a facilidade de implementação e a grande diversidade de padrões de redes de sensores suportados, devido ao fato das redes TCP/IP e a RSSF trabalharem isoladamente (KIM et al., 2007). No entanto, existem problemas relacionados à flexibilidade da solução, pois, nesse caso, geralmente existe um alto grau de acoplamento entre o protocolo da RSSF e a aplicação desenvolvida para interconectar as duas redes. Dessa forma, qualquer alteração na RSSF, como a

adição de um novo nó-sensor, por exemplo, implicará também numa modificação da aplicação do *gateway* ou *proxy* (WANG et al., 2009).

A Tabela 3 apresenta uma comparação entre os vários métodos de interconexão discutidos anteriormente.

Tabela 3: Comparação entre os diferentes tipos de interconexão.

Método	Custo	Acesso direto a um nó específico	Transparente para os usuários	Compatível com IPV4	Compatível com IPV6	Utiliza hardware extra
DTN	Alto	Não	Sim	Sim	Sim	Sim
WSN overlay TCP/IP	Alto	Sim	Não	Sim	Sim	Sim
TCP/IP overlay WSN	Alto	Sim	Sim	Sim	Sim	Sim
Virtual IP	Baixo	Sim	Sim	Sim	Sim	Sim
6LowPAN	Baixo	Sim	Sim	Não	Sim	Não
Gateway	Baixo	Não	Sim	Sim	Sim	Sim
Proxy	Baixo	Não	Sim	Sim	Sim	Sim

Fonte: Elaboração própria.

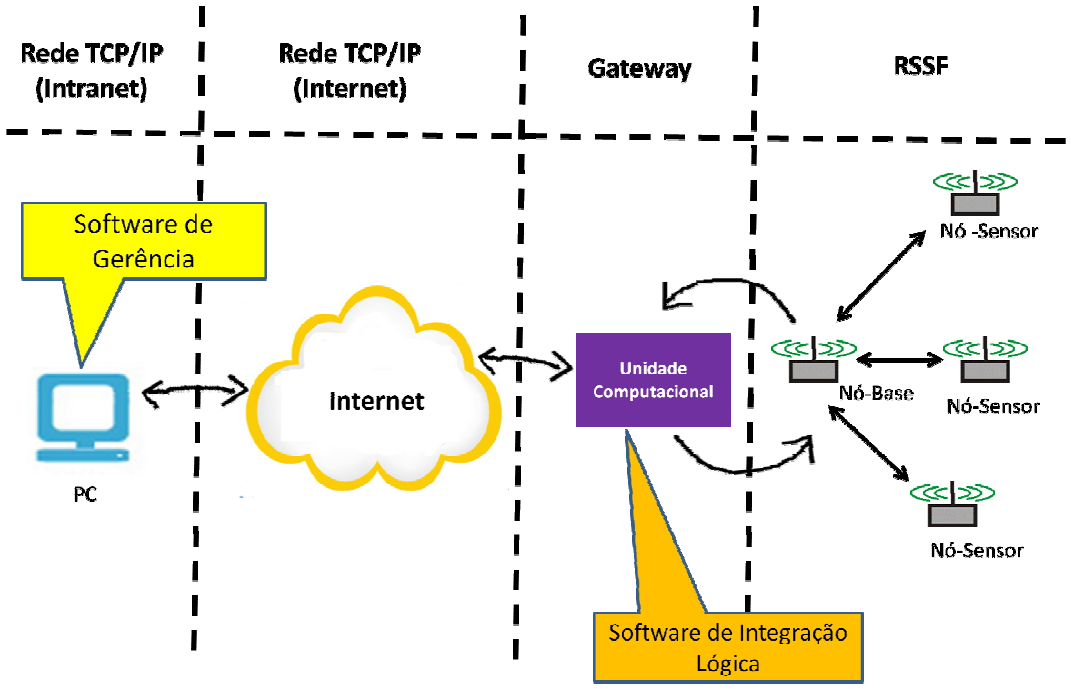
A proposta apresentada neste trabalho, busca uma solução de integração de RSSFs com redes TCP/IP somente utilizando a camada de aplicação, sem que haja a necessidade de alteração da pilha de protocolos de ambas as redes, preservando o funcionamento padrão de cada uma delas.

Os detalhes da solução proposta, serão discutidos no capítulo 5.

5. ARQUITETURA PARA INTERCONEXÃO DE RSSFS COM A INTERNET ATRAVÉS DO *PUBLISHER* HTTP

Normalmente nas propostas de interconexão que utilizam somente a camada de aplicação, o *software* de gerência fica localizado num *host* de uma rede TCP/IP. Um segundo *software*, que faz a interconexão lógica da RSSF com a rede TCP/IP, é instalado no *hardware* que conecta as duas redes fisicamente, conforme podemos observar na Figura 8. Geralmente, esse dispositivo de hardware é representado por um microcomputador ou outro dispositivo que possua uma capacidade mínima de processamento, memória e armazenamento para atender à aplicação que está sendo considerada.

Figura 8: Arquitetura de um sistema baseado na camada de aplicação.



Fonte: Elaboração própria.

Pode-se identificar nesse tipo de arquitetura de interconexão de RSSFs com redes TCP/IP, três componentes principais:

- O sistema de gerência, normalmente localizado num *host* da rede TCP/IP, representado por um microcomputador;

- O *hardware* que conecta as duas redes fisicamente, em geral, um microcomputador, mas podendo ser implementado com opções alternativas e de menor custo, como é o caso de um *access point* (MADEIRA, 2014) (CHEN et al., 2009);
- E por fim, o *software* responsável pela comunicação lógica entre a RSSF e a Rede TCP/IP.

5.1. Visão Geral da Arquitetura Proposta

A proposta apresentada por este trabalho, denominada *Publisher* HTTP, consiste em uma arquitetura para interconexão de RSSFs com a Internet, baseada no conceito de *proxy front-end server* (DUNKELS et al., 2004). Busca-se principalmente uma maior flexibilidade e escalabilidade do sistema, através de uma solução robusta e confiável, e que possa ser implementada com baixo custo.

Como se trata de uma solução de interconexão que se dá através de um software, atuante somente na camada de aplicação, optou-se pela utilização do protocolo HTTP como base da arquitetura pelos seguintes motivos:

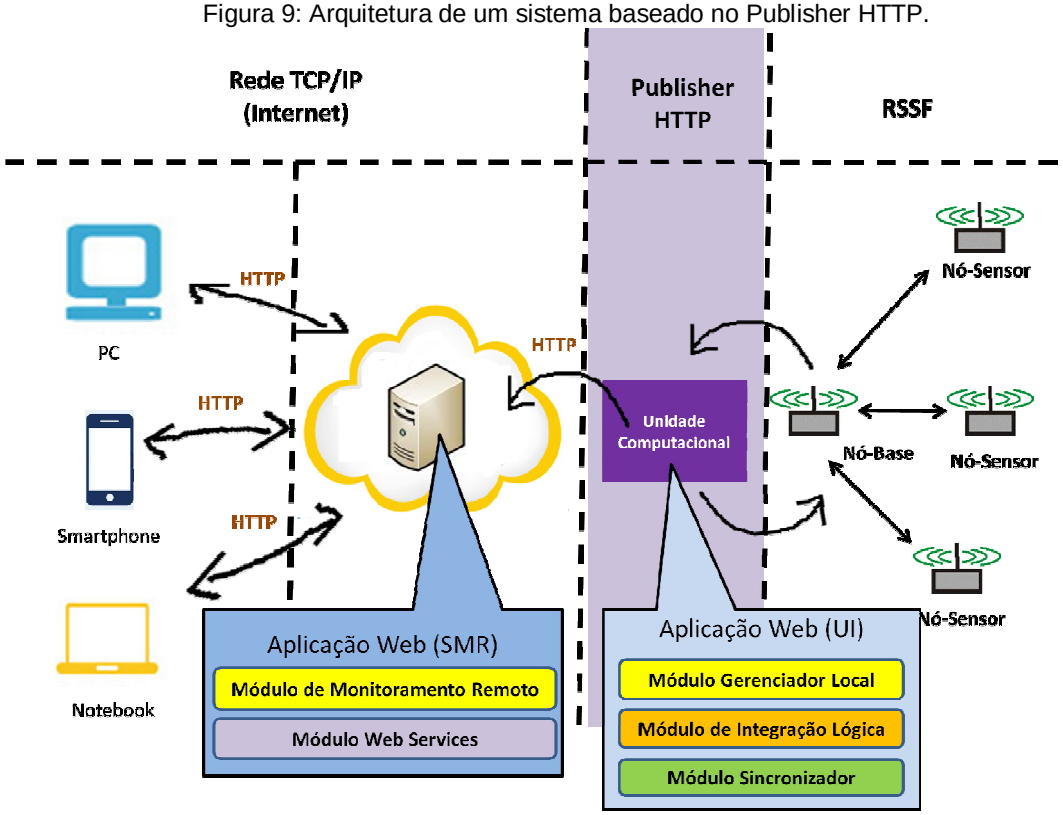
- É um protocolo da camada de aplicação;
- É comumente utilizado na Internet;
- Permite que a solução seja desenvolvida como uma aplicação *web*, tornando factível a integração da RSSF com a Internet através da tecnologia de *web services* suportada por uma estrutura de computação em nuvem.

Em comparação às propostas que conectam redes TCP/IP e RSSFs utilizando a camada de aplicação, como o que foi apresentado anteriormente, o Publisher HTTP apresenta os mesmos três elementos essenciais para que o sistema seja minimamente funcional: o sistema de gerência, o *hardware* que conecta as duas redes fisicamente e o *software* de integração lógica. No entanto, nessa arquitetura, o *softwares* de gerência e de integração lógica compõem uma única aplicação que será executada no próprio hardware do Publisher HTTP como uma aplicação *web* (ORACLE J2EE, 2005). Uma aplicação web é um *software* executado num servidor de aplicações ou container *web*, que recebe requisições, normalmente provenientes de um navegador *web*, executa um determinado

processamento no servidor, e retorna uma resposta ao cliente solicitante. Toda a comunicação entre essas partes é realizada com a utilização do protocolo HTTP.

Conforme podemos observar na Figura 9, a arquitetura conta ainda com um quarto elemento que corresponde a um sistema de monitoramento remoto, também uma aplicação do tipo *web*, suportada por uma estrutura de computação em nuvem, disponível na Internet.

Figura 9: Arquitetura de um sistema baseado no Publisher HTTP.



Fonte: Elaboração própria.

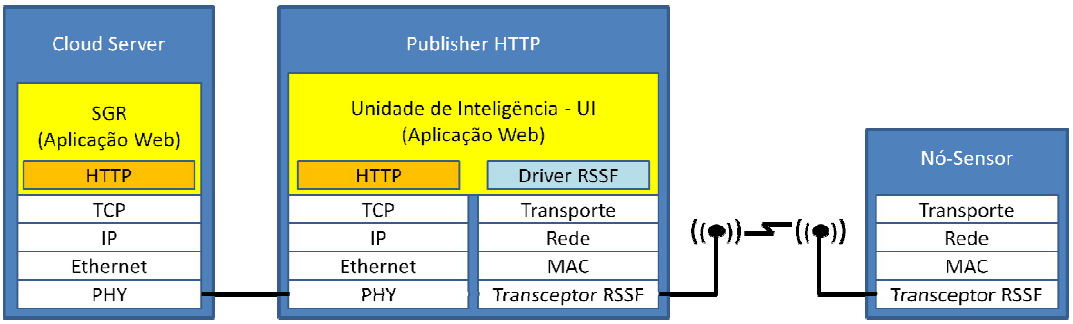
Conforme ilustrado na Figura 9, a arquitetura de um sistema baseado no *Publisher* HTTP pode ser dividida em quatro partes:

- a RSSF;
- o Publisher HTTP, que irá interconectar a RSSF à Internet;
- o Sistema de Monitoramento Remoto (SMR), responsável por disponibilizar as funções de monitoramento da RSSF na Internet;
- e os dispositivos e *softwares* clientes que poderão acessar as informações da RSSF através das funções disponibilizadas pelo SMR.

As duas partes que suportam essa arquitetura, e que são o foco deste trabalho, são o SMR e o *Publisher* HTTP, os quais serão detalhados a seguir.

A Figura 10 exibe uma visão geral da arquitetura proposta, com base na comunicação entre as pilhas de protocolos da rede TCP/IP e da RSSF.

Figura 10: Comunicação entre as pilhas de protocolos das duas redes.



Fonte: Elaboração própria.

5.2. O *Publisher* HTTP

O nome utilizado para esse componente, *Publisher*, refere-se ao termo publicador, tradução do termo originalmente da língua inglesa, para o português. A escolha desse termo deve-se à função desse componente que, ao enviar dados da RSSF para a Internet, estará, de certa forma, publicando tais informações na Internet.

A principal função do *Publisher* HTTP é fazer com que as informações provenientes da RSSF fiquem disponíveis na Internet. Este componente do sistema cumpre um importante papel na interconexão da RSSF com a Internet, através de uma aplicação *web* instalada num dispositivo de *hardware* que conecta as duas redes fisicamente. Ou seja, o Publisher HTTP é formado tanto por componentes de *hardware* quanto de *software*.

Em termos de *hardware* o *Publisher* HTTP é composto somente pela Unidade Computacional (UC). A UC é um hardware que deve ser dotado de capacidade de processamento e armazenamento de informações, necessários para que a integração entre a RSSF e a Internet seja possível. Este componente deve estar

conectado fisicamente à Internet e à RSSF, e deve utilizar um sistema operacional para permitir o controle e funcionamento de todos os seus recursos.

A porção inteligente do *Publisher* HTTP está concentrada na aplicação *web* denominada Unidade de Inteligência (UI), a qual corresponde ao componente de *software* do *Publisher* HTTP. A UI deve estar instalada na UC, e está subdividida em três módulos, o Módulo Gerenciador Local (MGL), o Módulo de Integração Lógica (MIL) e o Módulo Publicador (MP). É importante ressaltar a necessidade de um servidor web e um banco de dados relacional instalados na UC, para que a aplicação web seja executada e possa armazenar tantos as informações de configuração do sistema, quanto aquelas necessárias à gerência da RSSF e dos dados coletados por ela.

O Módulo Gerenciador Local (MGL) é a parte da aplicação web que deve prover todas as funções de gerência como configuração, monitoramento e controle, tanto dos dados coletados pela RSSF quanto das informações técnicas de funcionamento da rede. Esse módulo deve oferecer ao administrador do sistema a possibilidade de configurar uma rede de sensores sem fio, seus nós-sensores e as informações de cada nó-sensor que devem ser monitoradas e controladas.

O MGL pode ser acessado através de uma interface web tanto para a realização da configuração da RSSF e dos parâmetros que serão monitorados, quanto para o seu posterior monitoramento. No entanto, isso só será possível quando o administrador estiver conectado à mesma intranet TCP/IP onde o Publisher HTTP se encontra localizado.

Diferentemente do módulo anterior, o Módulo de Integração Lógica (MIL), não necessita de uma interface com o usuário. Esse módulo é responsável pela comunicação entre a Internet e a RSSF, recebendo solicitações do MGL, traduzindo tais solicitações para o protocolo da RSSF, e enviando-as para a rede de sensores através de um driver específico para o padrão da RSSF que está sendo utilizado. Quando o MIL recebe a resposta da solicitação, vinda da RSSF, ele executa a operação inversa, obtendo as informações desejadas a partir do protocolo da RSSF e armazenando-as no banco de dados local do Publisher HTTP. É importante ressaltar a necessidade de que o MIL seja programado de forma a permitir a

comunicação com RSSFs de diferentes padrões, pois isso torna o sistema mais flexível e escalável, e reduz o grau de acoplamento com um protocolo específico de RSSF.

Para finalizar, temos o Módulo Publicador (MP). Esse módulo é responsável por manter as informações contidas no Sistema de Monitoramento Remoto (SMR) atualizadas em relação àquelas armazenadas no *Publisher* HTTP. Ou seja, os dados captados pela RSSF devem ser replicados do *Publisher* HTTP para o SMR periodicamente. Dessa forma, o MP deve garantir que todos os dados presentes no bancos de dados do MGL sejam enviados para o SMR. Por exemplo, se um novo nó-sensor é adicionado ao sistema ou o intervalo de tempo de leitura é alterado no MGL, essas informações também devem ser publicadas no SMR, além dos dados de monitoramento.

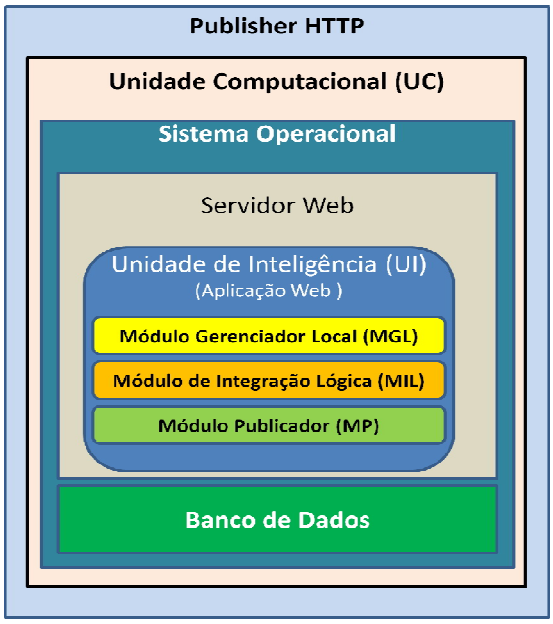
Essa função, de envio dos dados do *Publisher* HTTP para o SMR, deve ser realizada com a utilização de web services. Nesse contexto, o *Publisher* HTTP atuará como uma aplicação cliente do *web service* localizado no SMR, e preparado para receber tais informações. Dessa forma, pode-se dizer que, de acordo com a arquitetura proposta, a integração entre a RSSF e a Internet através da tecnologia de web services, exige a utilização dessas duas partes, e o *Publisher* HTTP tem um importante papel na realização das tarefas referentes ao lado cliente dessa integração.

Também é de responsabilidade do MP, uma vez que os dados das grandezas monitoradas pela RSSF são replicados para o SMR, efetuar a remoção dessas informações da base de dados do Publisher HTTP, de tempos em tempos. Dessa forma, o sistema pode manter um nível de performance elevado. A peridiocidade com que essa operação deve ocorrer pode ser configurada no MGL, e fica a critério do administrador do sistema defini-la.

A ideia, portanto, é manter o histórico das informações captadas pela RSSF a longo prazo, no SMR. Pois, ele se encontra num ambiente de computação em nuvem e não corre o risco de ser impactado pela falta de recursos. No Publisher HTTP, devem ser mantidas somente as informações mais recentes.

A Figura 11 apresenta uma visão geral da arquitetura do *Publisher* HTTP.

Figura 11: Arquitetura do *Publisher* HTTP.



Fonte: Elaboração própria.

É importante destacar que cada *Publisher* HTTP é capaz de conectar apenas uma RSSF à Internet.

5.3. O Sistema de Monitoramento Remoto (SMR)

Para estender as funcionalidades de monitoramento do sistema para o contexto da Internet é que foi criado o componente SMR na arquitetura proposta.

O *Publisher* HTTP faz a publicação dos seus dados locais para um servidor suportado por uma estrutura de computação em nuvem, disponível na Internet.

Nesse servidor, temos um sistema de monitoramento, também uma aplicação do tipo *web*, denominado Sistema de Monitoramento Remoto (SMR), que pode ser subdividido em dois módulos: o Módulo de Monitoramento Remoto (MMR) e o Módulo *Web Services* (MWS).

O Módulo de Monitoramento Remoto (MMR) possui basicamente as mesmas funções de monitoramento disponibilizadas pelo componente MGL do *Publisher* HTTP, mas que podem ser acessadas via Internet. Através do MMR o administrador do sistema poderá efetuar o monitoramento sistema do independentemente da sua localização física, desde que tenha acesso à Internet, e através de qualquer

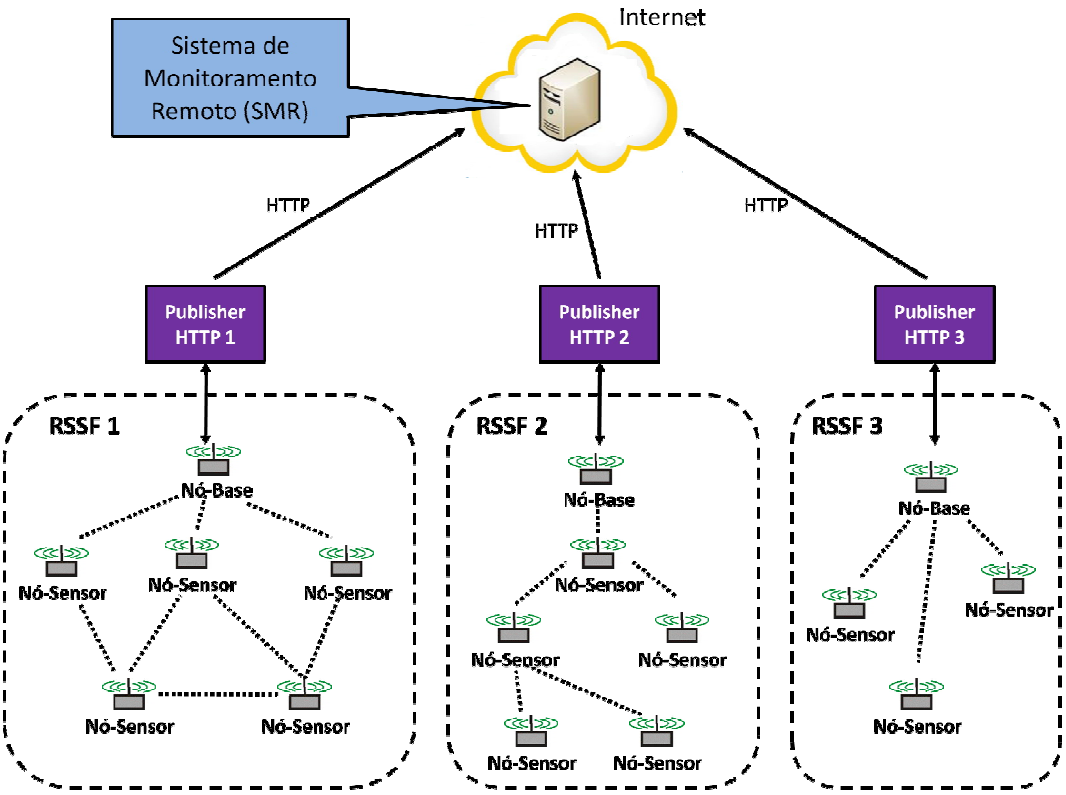
dispositivo de hardware que possa fazer uso de um navegador *web*, como um microcomputador, um *tablet* ou mesmo um *smartphone*.

A principal função do Módulo *Web Services* (MWS) é disponibilizar o web service responsável por receber as informações provenientes do *Publisher* HTTP. Ou seja, esse web service é essencial para que a integração ocorra, assim como é essencial que o *Publisher* HTTP atue como um cliente desse *web service*, enviando as informações armazenadas em seu banco de dados local. Além disso, o MWS é responsável por disponibilizar as funções monitoramento do MMR por meio da tecnologia de *web services*. Isto permite a integração SMR com outros sistemas compatíveis com a mesma tecnologia, criando independência em relação às plataformas de *hardware* e *software* utilizadas, e proporcionando a utilização das informações provenientes da RSSF por aplicações móveis e por sistemas desenvolvidos por terceiros.

Como já foi citado anteriormente, o propósito do SMR, além de disponibilizar as funções de monitoramento da RSSF via Internet, é o de manter todo o histórico dos dados capturados a longo prazo, de forma que o *Publisher* HTTP necessite armazenar essas informações o mínimo de tempo necessário. O objetivo é o não comprometimento dos recursos de hardware do *Publisher* HTTP, para que ele possa manter um nível de performance elevado e para que não haja impactos negativos no funcionamento do sistema.

O SMR pode monitorar tantas RSSFs quanto lhe for capaz, independentemente do padrão da RSSF, do número de nós-sensores e da sua topologia, desde que a plataforma utilizada seja suportada pelo *Publisher* HTTP, conforme mostra a Figura 12. Portanto, a sua utilização dentro de uma infra-estrutura de computação em nuvem é essencial para tornar o sistema ainda mais escalável e flexível, pois, numa estrutura como essas, os recursos como memória, processamento e armazenamento, podem ser adicionados conforme a necessidade e com muita facilidade, permitindo que o sistema mantenha uma boa performance mesmo que esteja crescendo.

Figura 12: Gerenciamento de várias RSSFs centralizado no SGR.



Fonte: Elaboração própria.

Obviamente, um esquema de monitoramento centralizado como o exibido na Figura 12, só é justificável quando temos as diferentes RSSFs sendo utilizadas para uma mesma finalidade ou aplicação, como por exemplo, o gerenciamento de estufas utilizadas para o plantio de frutas e vegetais em ambientes fechados como alface, uva e morango (BENAVENTE, 2010; ALVES, 2011). Nesse caso, um produtor que tenha que monitorar o ambiente de estufas que se encontram fisicamente distantes umas das outras, verificando variáveis como temperatura, luminosidade, umidade relativa do ar, etc., contaria com a praticidade de poder efetuar o monitoramento dessas estufas através de um único sistema gerenciador, no caso, o SMR.

6. MATERIAIS E MÉTODOS

Neste capítulo serão apresentados detalhes da implementação da proposta apresentada, integrando uma RSSF do padrão Radiuino, a um servidor localizado na Internet com a utilização da tecnologia de *web services* e suportado por uma estrutura de computação em nuvem.

Objetivando demonstrar que, tanto o *Publisher* HTTP quanto o sistema como um todo, tiveram como premissa uma implementação alternativa e de baixo custo, foram levados em consideração somente opções de hardware e software que se enquadrassem nas categorias de *open-hardware* e *open-source*.

6.1. O *Hardware* do *Publisher* HTTP

O *hardware* da Unidade Computacional (UC) foi definido com base nos seguintes requisitos:

- capacidade de processamento e armazenamento;
- custo;
- tamanho físico;
- comunicação Ethernet e USB;
- quantidade de memória RAM.

Buscou-se avaliar opções que se enquadrassem na categoria open-hardware, com custo inferior a 50 dólares e um tamanho físico suficientemente reduzido para ser utilizado em locais como pequenas estufas voltadas para o plantio de frutas e hortaliças. Três mini-computadores foram selecionados: Rasperry Pi (RASPERRY PI, 2014), BeagleBone Black (BEAGLEBOARD, 2014), Arduino Uno (ARDUINO, 2014).

A Tabela 4 apresentada as características que foram consideradas importantes para cada uma das opções de *hardware* avaliadas.

Tabela 4: Comparação entre Beaglebone Black, Raspberry Pi e Arduino Uno.

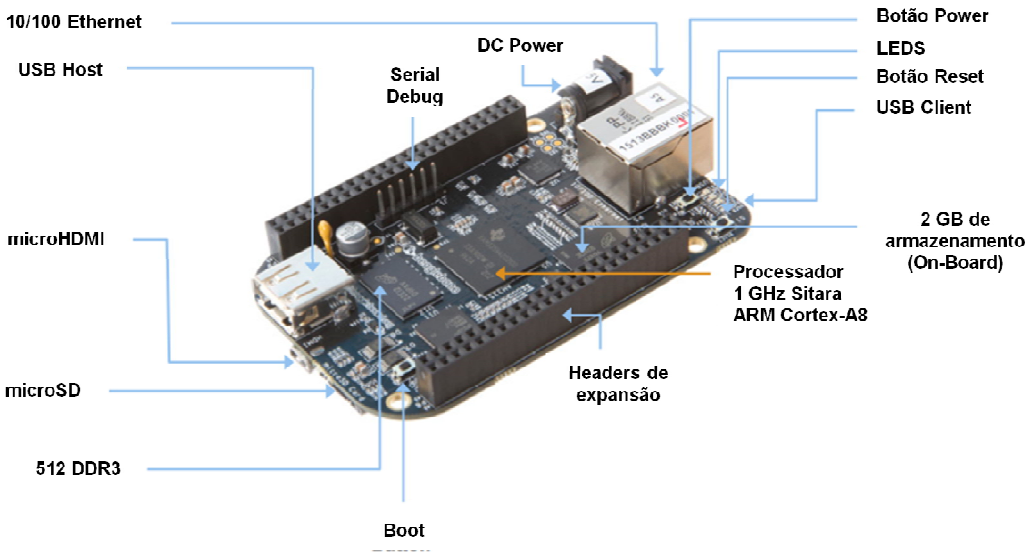
	Raspberry Pi - Model B	Arduino Uno	BeagleBone Black
Custo	35 dólares	35 dólares	45 dólares
Processador	ARM1176JZFS - Broadcom BCM2835	AVR - AtMega 328	Texas Instruments AM3359 - ARM Cortex A8
Velocidade de processamento	700MHz	16MHz	1GHz
RAM	512MB	2KB	512MB
Armazenamento	SD Card	32KB	2GB + Micro SD Card
USB	2 Host	1 Host	1 Host, 1 Client
Ethernet	10/100 Mbs	Não possui	10/100 Mbs
Tamanho físico	8.6 x 5.4 x 1.7cm	7.6 x 6.4 x 1.9cm	8.6 x 5.33 x 0.5cm
Tensão de Aliemntação:	5 Volts	5 Volts	5 Volts

Fonte: Elaboração própria.

O BeagleBone Black foi a opção escolhida por apresentar características que o colocam num patamar muito acima das demais opções avaliadas, demonstrando ser muito poderoso e economicamente acessível. Mesmo possuindo um custo superior aos 35 dólares das outras duas opções de hardware consideradas, o BeagleBone Black justifica seu custo de 45 dólares por sua superioridade comprovada em vários aspectos. Em relação ao Arduino Uno, o BeagleBone Black possui um processador milhares de vezes mais rápido, e além de apresentar um processamento superior também em relação ao Rasperry Pi, o BeagleBone Black apresenta mais interfaces de entrada e saída e maior capacidade de armazenamento.

A Figura 13 exibe uma foto do mini-computador BeagleBone Black destacando alguns de seus componentes.

Figura 13: BeagleBone Black e seus componentes



Fonte: (BEAGLEBOARD, 2014).

A utilização deste hardware, buscando atender aos requisitos expostos anteriormente, juntamente com a necessidade de suporte a um banco de dados relacional e um servidor para aplicações web, orientaram para a escolha de um sistema operacional baseado em Linux. Entre as distribuições Linux disponíveis para BeagleBone Black optou-se pela utilização do Linux Ubuntu 3.8.13, desenvolvido para dispositivos baseados na arquitetura Advanced RISC Machine - ARM, uma vez que esta é uma indicação do próprio projeto BeagleBoard, além de ser uma distribuição Linux já consolidada, há cerca de nove anos no mercado.

6.2. O *Software* da Solução Proposta

Inicialmente, a decisão sobre qual linguagem de programação deveria ser utilizada para o desenvolvimento do software da solução proposta, foi pautada pelos seguintes itens já citados anteriormente:

- suporte ao desenvolvimento de aplicações do tipo *web*;
- compatibilidade com a tecnologia de *web services*;
- suporte ao desenvolvimento de baixo-nível, como manipulação dos recursos de *hardware*;
- robustez e escalabilidade;

- processamento concorrente;
- compatibilidade com o sistema operacional Linux.

Baseado em tais fatores optou-se pela utilização da linguagem de programação Java (ORACLE JAVA, 2015) em relação à linguagem PHP (PHP, 2015), por se mostrar uma opção mais robusta e com melhor suporte ao desenvolvimento de baixo-nível.

A tabela 5, faz uma comparação das características levadas em consideração nessa avaliação. As informações foram obtidas a partir do web site oficial de ambas as linguagens.

Tabela 5: Comparação entre Java e PHP.

Recurso	Java	PHP
Orientação a objetos	Suporte nativo	Suporta parcialmente
Coletor de lixo (Memória)	Suporte nativo	Não suporta
Caching de objetos	Suporte nativo	Não suporta
Multi-plataforma (hardware e software - Desktop, Mobile, Web, etc.)	Suporte nativo	Não suporta
Suporte a operação de debug	Suporte nativo	Não suporta
Threads (Processamento paralelo)	Suporte nativo	Não suporta
Mecanismo de persistência	Suporte nativo	Suportado por API's de terceiros
Mecanismo de transação	Suporte nativo	Não suporta
API de gerenciamento de aplicações	Suporte nativo	Não suporta

Fonte: (ORACLE JAVA, 2015) (PHP, 2015).

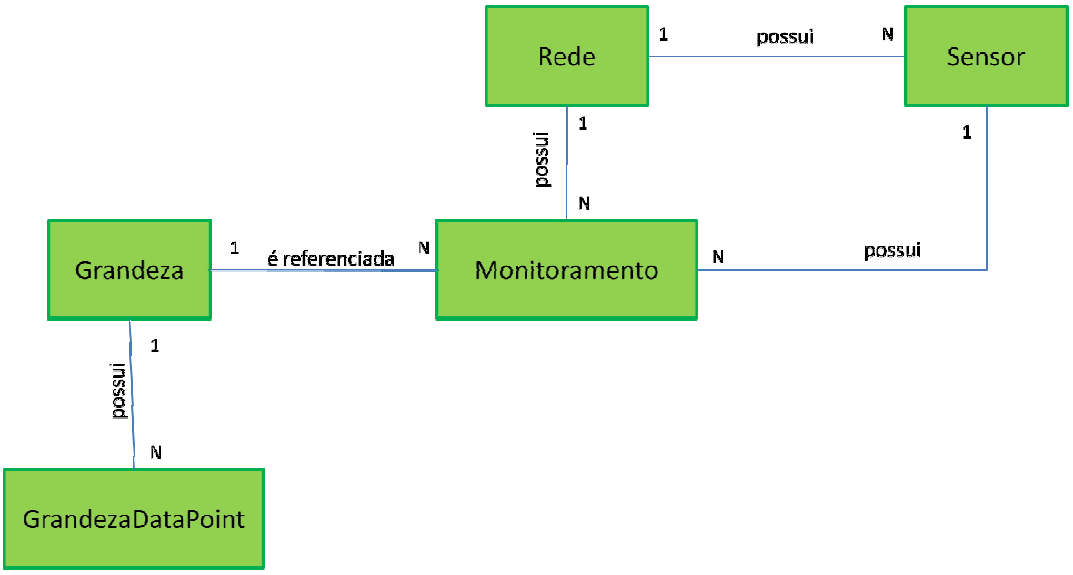
6.2.1. O Software do *Publisher* HTTP

A parte do *Publisher* HTTP referente a software, definida como Unidade de Inteligência (UI), foi desenvolvida como uma aplicação Java Web, subdividida em três módulos, conforme apresentado pela arquitetura proposta: o Módulo Gerenciador Local (MGL), o Módulo de Integração Lógica e o Módulo Publicador.

6.2.1.1. A base de dados do Publisher HTTP

Para suportar os módulos que compõem a UI, foi definido um banco de dados, conforme a estrutura exibida no diagrama de entidade-relacionamento da Figura 14. Esse diagrama mostra as tabelas criadas para esse banco de dados e como elas se relacionam entre si. Cada retângulo verde representa uma tabela, e uma linha conectando um retângulo a outro, indica que essas tabelas se relacionam. Os verbos sobre as linhas demonstra como o relacionamento deve ser compreendido, e os números e letras nas extremidades, indicam a cardinalidade máxima. Por exemplo, uma rede pode possuir muitos sensores, mas um sensor pertence somente a uma rede. Um sensor pode ser referenciado por muitos monitoramentos, mas um monitoramento específico refere-se somente a um único sensor.

Figura 14: Estrutura do banco de dados que suporta os módulos da UI.



Fonte: Elaboração própria.

Esse banco de dados irá armazenar tanto informações de configuração da RSSF, quanto das leituras realizadas para cada grandeza monitorada.

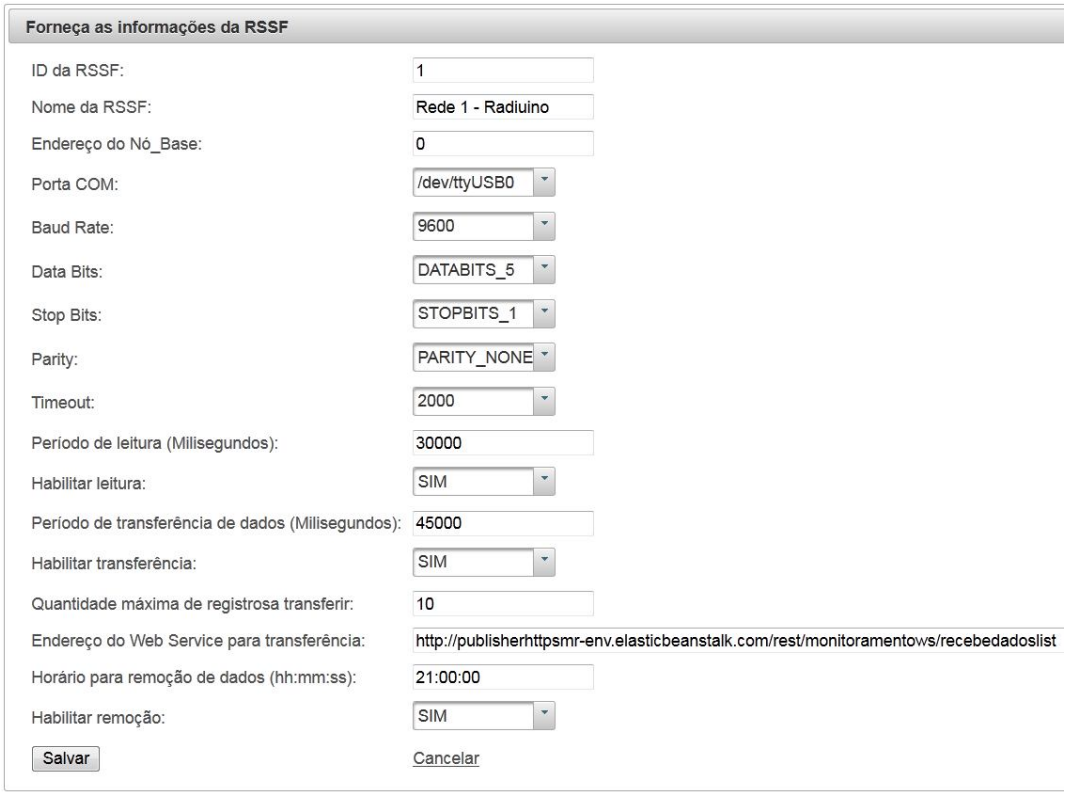
A descrição de cada uma das tabelas que compõem o banco de dados *Publisher* HTTP e seus respectivos campos se encontra no Apêndice A.

6.2.1.2. O Módulo Gerenciador Local (MGL)

O MGL consiste na parte do *Publisher* HTTP responsável pela interface com usuário. Esse módulo disponibiliza todas as funções para que o sistema seja devidamente configurado e colocado em operação.

Através do MGL deve-se fornecer informações tais como nome da RSSF, endereço lógico do nó-base, porta serial utilizada pelo nó-base, período de leitura, dentre outras informações essenciais ao funcionamento do sistema conforme mostra a Figura 15.

Figura 15: Interface para cadastro da RSSF.



Fonte: Elaboração própria.

Além das informações básicas da RSSF, deve-se utilizar o MGL para efetuar o cadastro dos nós-sensores que compõem a RSSF, informando o endereço lógico de cada nó-sensor, e das grandezas que se deseja monitorar. No caso das grandezas, deve-se especificar no ato do cadastro, quais *bytes* do pacote de dados recebido dos nós-sensores irá compor cada uma delas. Tais detalhes sobre o

pacote de dados utilizado nessa dissertação, serão discutidos no próximo tópico deste capítulo.

Todas essas informações relacionadas à configuração da RSSF, dos nós-sensores e das grandezas monitoradas, são armazenadas pelo MGL no banco de dados local, mais especificamente nas tabelas Rede, Sensor, Grandeza e.

O MGL conta ainda com funções para alteração e exclusão de informações, como mostra a Figura 16, para garantir a manutenção completa do sistema.

Figura 16: Funções de exclusão e alteração dos componentes do sistema.

Temperatura

Indice do Byte	Identificação do Byte	Ações	Ações
16	AD0_W	Excluir	Atualizar
17	AD0_H	Excluir	Atualizar
18	AD0_L	Excluir	Atualizar

Fonte: Elaboração própria.

As funções de monitoramento estão relacionadas à recursos destinados à geração de listagens dos componentes do sistema e relatórios com base nos dados coletados da RSSF, conforme pode-se observar na Figura 17.

Figura 17: Relatório de temperatura por período

Informações de monitoramento

Filtro e ordenação

05/14/2015 11:11:00 | 05/14/2015 11:16:00 | TEMPERATURA | Confirmar | Limpar

Solicitação	Retorno	Transferência Web	Valor
2015-05-14 11:11:12.0	2015-05-14 11:11:14.0	2015-05-14 11:11:17.0	22.740
2015-05-14 11:11:32.0	2015-05-14 11:11:34.0	2015-05-14 11:11:52.0	22.740
2015-05-14 11:12:02.0	2015-05-14 11:12:04.0	2015-05-14 11:12:07.0	23.063
2015-05-14 11:12:32.0	2015-05-14 11:12:34.0	2015-05-14 11:12:52.0	23.063
2015-05-14 11:13:02.0	2015-05-14 11:13:04.0	2015-05-14 11:13:07.0	23.063
2015-05-14 11:13:32.0	2015-05-14 11:13:34.0	2015-05-14 11:13:52.0	23.385
2015-05-14 11:14:02.0	2015-05-14 11:14:04.0	2015-05-14 11:14:07.0	23.063
2015-05-14 11:14:32.0	2015-05-14 11:14:34.0	2015-05-14 11:14:52.0	23.063
2015-05-14 11:15:02.0	2015-05-14 11:15:04.0	2015-05-14 11:15:07.0	23.063
2015-05-14 11:15:32.0	2015-05-14 11:15:34.0	2015-05-14 11:15:52.0	23.063

Voltar

Fonte: Elaboração própria.

6.2.1.3. O Módulo de Integração Lógica (MIL)

O Módulo de Integração Lógica (MIL) é o componente que efetivamente faz a comunicação entre o Publisher HTTP e a RSSF, programaticamente.

Utilizando as informações de configuração adicionadas ao sistema através do MGL, o MIL estabelecerá uma comunicação com a RSSF, enviando solicitações de leituras das grandezas monitoradas aos nós-sensores que compõem a rede. Ao receber a resposta da solicitação efetuada de cada nó-sensor, o MIL efetua os cálculos necessários para obtenção dos valores das grandezas monitoradas, e armazena tais dados na tabela Monitoramento do banco de dados local.

Mais especificamente, o MIL monta um pacote de dados com as informações necessárias para a solicitação que se deseja realizar, e envia-o para o nó-base da RSSF através da porta serial do hardware utilizado. O nó-base envia o pacote de solicitação ao nó-sensor em questão, recebe outro pacote em resposta, repassando-o novamente ao MIL.

Ao receber o pacote de resposta, o MIL obtém os valores desejados e os armazena no banco de dados do sistema. A estrutura do pacote de dados utilizado para envio e recebimento das informações aos nós-sensores, está intimamente ligada ao padrão da RSSF com o qual se está trabalhando.

No caso desse experimento o padrão escolhido, foi a plataforma open source Radiuino (RADIUINO, 2014). O Radiuino utiliza um pacote de dados formado por 52 *Bytes*, divididos em 16 bytes para os cabeçalhos das camadas Física, MAC, Rede e Transporte, e 36 bytes destinados à camada de aplicação. Desses 36 bytes, 18 são destinados aos conversores analógico-digitais, e 18, às entradas e saídas digitais. Cada byte possui uma posição e uma função especifica dentro do pacote, armazenando determinado valor de acordo com sua finalidade. A Figura 18 mostra o mapa de bytes do pacote Radiuino.

Figura 18: Mapa de bytes do pacote Radiuino.

Cabeçalhos das Camadas	PhyHdr[#]				MacHdr[#]			
Posição na variável	0	1	2	3	0	1	2	3
Função das posições do cabeçalho	RSSI Donwlink	LQI Donwlink	RSSI Uplink	LQI Uplink	Sleep	Tempo Sleep 1	Tempo Sleep 2	TDB
Posição dos bytes no pacote	0	1	2	3	4	5	6	7

Cabeçalhos das Camadas	NetHdr[#]				TranspHdr[#]			
Posição na variável	0	1	2	3	0	1	2	3
Função das posições do cabeçalho	DST_ID Quem Recebe	DST_NID	SRC_ID Quem Manda	SRC_NID	COUNT	TDB	TDB	TDB
Posição dos bytes no pacote	8	9	10	11	12	13	14	15

Conversores AD	AD0[#]			AD1[#]			AD2[#]		
Posição na variável dos AD	0	1	2	0	1	2	0	1	2
Função das posições dos AD	AD0[0]	AD0[1]	AD0[2]	AD1[0]	AD1[1]	AD1[2]	AD2[0]	AD2[1]	AD2[2]
Posição dos bytes no pacote	16	17	18	19	20	21	22	23	24

Conversores AD	AD3[#]			AD4[#]			AD5[#]		
Posição na variável dos AD	0	1	2	0	1	2	0	1	2
Função das posições dos AD	AD3[0]	AD3[1]	AD3[2]	AD4[0]	AD4[1]	AD4[2]	AD5[0]	AD5[1]	AD5[2]
Posição dos bytes no pacote	25	26	27	28	29	30	31	32	33

Entradas e saídas digitais	IO0[#]			IO1[#]			IO2[#]		
Posição na variável dos IO	0	1	2	0	1	2	0	1	2
Função das posições dos IO	IO0[0]	IO0[1]	IO0[2]	IO1[0]	IO1[1]	IO1[2]	IO2[0]	IO2[1]	IO2[2]
Posição dos bytes no pacote	34	35	36	37	38	39	40	41	42
Entradas e saídas digitais	IO3[#]			IO4[#]			IO5[#]		
Posição na variável dos IO	0	1	2	0	1	2	0	1	2
Função das posições dos IO	IO3[0]	IO3[1]	IO3[2]	IO4[0]	IO4[1]	IO4[2]	IO5[0]	IO5[1]	IO5[2]
Posição dos bytes no pacote	43	44	45	46	47	48	49	50	51

Fonte: Modificado de (RADIUINO, 2014).

A Tabela 6 exibe uma explicação mais detalhada dos campos que formam o pacote de dados utilizado pela plataforma Radiuino:

Tabela 6: Detalhamento do pacote Radiuino.

Camada	Descrição
Física (PHY)	Conjunto de 4 bytes onde se encontram as informações de potência de recepção do sinal RSSI_DLINK e RSSI_ULINK e os indicadores de qualidade do sinal LQI_DLINK e LQI_ULINK.
Controle de Acesso ao Meio (MAC)	Conjunto de 4 bytes destinados à implementação de regras para acesso ao meio físico utilizado.
Rede (Net)	Conjunto de 4 bytes onde se encontram os endereços de origem e destino do pacote.
Transporte (Transp)	Conjunto de 4 bytes dividido em 1 byte utilizado como contador de pacotes e 3 bytes de uso geral.
Aplicação	AD0 a AD5: Conjunto de 18 bytes onde estão disponíveis os bytes inferior (ADx_L) e superior (ADx_H) dos conversores analógico-digitais e 1 byte de configuração. O Radiuino conta com 6 conversores analógico-digitais e cada um deles utiliza 3 bytes.
	IO0 a IO5: Conjunto de 18 bytes onde estão disponíveis os bytes inferior (ADx_L) e superior (ADx_H) das entradas e saídas digitais e um terceiro byte para uso geral. O Radiuino conta com 6 entradas e saídas digitais e cada uma delas utiliza 3 bytes.

Fonte: Elaboração própria.

Para esse experimento, de acordo com o firmware instalado previamente no nó-base e no nó-sensor da RSSF, foram considerados os campos do pacote Radiuino utilizados para o monitoramento de RSSI, temperatura e luminosidade. A Tabela 7 exibe em detalhes, cada byte do pacote Radiuino utilizado por este experimento.

Tabela 7: Detalhamento do pacote Radiuino.

Grandeza monitorada	Cabeçalho	Função	Número do byte no pacote
RSSI	PHYHdr[#]	RSSI Downlink	0
	PHYHdr[#]	RSSI Uplink	2
Temperatura	AD0H[#]	AD0_W	16
	AD0H[#]	AD0_H	17
	AD0H[#]	AD0_L	18
Luminosidade	AD1H[#]	AD1_W	19
	AD1H[#]	AD1_H	20
	AD1H[#]	AD1_L	21

Fonte: Elaboração própria.

Após obter os dados provenientes do pacote Radiuino, o MIL efetua o cálculo das grandezas monitoradas: RSSI, temperatura e luminosidade. A Figura 19 exibe o código-fonte Java utilizado para tais cálculos.

Figura 19: Código-fonte Java para cálculo do RSSI, temperatura e luminosidade.

Radiuino.java

```
    public Double calculaTemperatura(int ad0h, int ad0l) {

        int AD0 = (ad0h * 256 + ad0l);
        Double vout = 0.003223 * AD0;
        Double temp = (vout * 100) - 53;

        return temp;

    }

    public Double calculaLuminosidade(int ad0h, int ad0l) {

        Double luminosidade = new Double(ad0h * 256 + ad0l);

        return luminosidade;

    }

    public Double calculaRssiUp(int rssiUp) {

        Double rssiUpCalc;

        if (rssiUp > 128) {
            rssiUpCalc = ((rssiUp - 256) / 2.0) - 74;
        } else {
            rssiUpCalc = (rssiUp / 2.0) - 74;
        }

        return rssiUpCalc;

    }

    public Double calculaRssiDown(int rssiDown) {

        Double rssiDownCalc;

        if (rssiDown > 128) {
            rssiDownCalc = ((rssiDown - 256) / 2.0) - 74;
        } else {
            rssiDownCalc = (rssiDown / 2.0) - 74;
        }

        return rssiDownCalc;

    }
}
```

Fonte: Elaboração própria.

Os dados calculados são persistidos na tabela Monitoramento do banco de dados do sistema, conforme pode-se observar na Figura 20, um conjunto de registros relacionados ao monitoramento da temperatura.

Figura 20: Dados gravados nas tabelas Monitoramento.

Status | Result1

	IDREDE	IDNOSENSOR	IDGRANDEZA	DATAHORASOLICITACAORSSF	DATAHORARETORNORSSF	VALORGRANDEZA
1	1	1	1	2015-05-14 11:11:12.0	2015-05-14 11:11:14.0	22.740499999999997
2	1	1	1	2015-05-14 11:11:32.0	2015-05-14 11:11:34.0	22.740499999999997
3	1	1	1	2015-05-14 11:12:02.0	2015-05-14 11:12:04.0	23.06280000000001
4	1	1	1	2015-05-14 11:12:32.0	2015-05-14 11:12:34.0	23.06280000000001
5	1	1	1	2015-05-14 11:13:02.0	2015-05-14 11:13:04.0	23.06280000000001
6	1	1	1	2015-05-14 11:13:32.0	2015-05-14 11:13:34.0	23.38510000000001
7	1	1	1	2015-05-14 11:14:02.0	2015-05-14 11:14:04.0	23.06280000000001
8	1	1	1	2015-05-14 11:14:32.0	2015-05-14 11:14:34.0	23.06280000000001

Fonte: Elaboração própria.

6.2.1.4. O Módulo Publicador (MP)

Para finalizar, temos o Módulo Publicador (MP). Esse módulo é responsável por manter as informações contidas no Sistema de Monitoramento Remoto (SMR) atualizadas em relação àquelas armazenadas no *Publisher* HTTP. Ou seja, os dados captados pela RSSF devem ser replicados do *Publisher* HTTP para o SMR periodicamente. Dessa forma, o MP deve garantir que todos os dados presentes no bancos de dados do MGL sejam enviados para o SMR. Por exemplo, se um novo nó-sensor é adicionado ao sistema ou o intervalo de tempo de leitura é alterado no MGL, essas informações também devem ser publicadas no SMR, além dos dados de monitoramento.

Também é de responsabilidade do MP, uma vez que os dados das grandezas monitoradas pela RSSF são replicados para o SMR, efetuar a remoção dessas informações da base de dados do Publisher HTTP, de tempos em tempos. Dessa forma, o sistema pode manter um nível de performance elevado. A peridiocidade com que essa operação deve ocorrer pode ser configurada no MGL, e fica a critério do administrador do sistema defini-la.

O Módulo Publicador (MP), tem a função de manter as informações contidas no Sistema de Monitoramento Remoto (SMR), atualizadas em relação àquelas armazenadas no banco de dados do *Publisher* HTTP.

Essas informações incluem tanto aquelas referentes à configuração da RSSF, como por exemplo, intervalo de leitura das grandezas monitoradas e nós-sensores que compõem a rede, quanto os dados captados pelos nós-sensores e as informações geradas a partir deles, como a temperatura e a luminosidade.

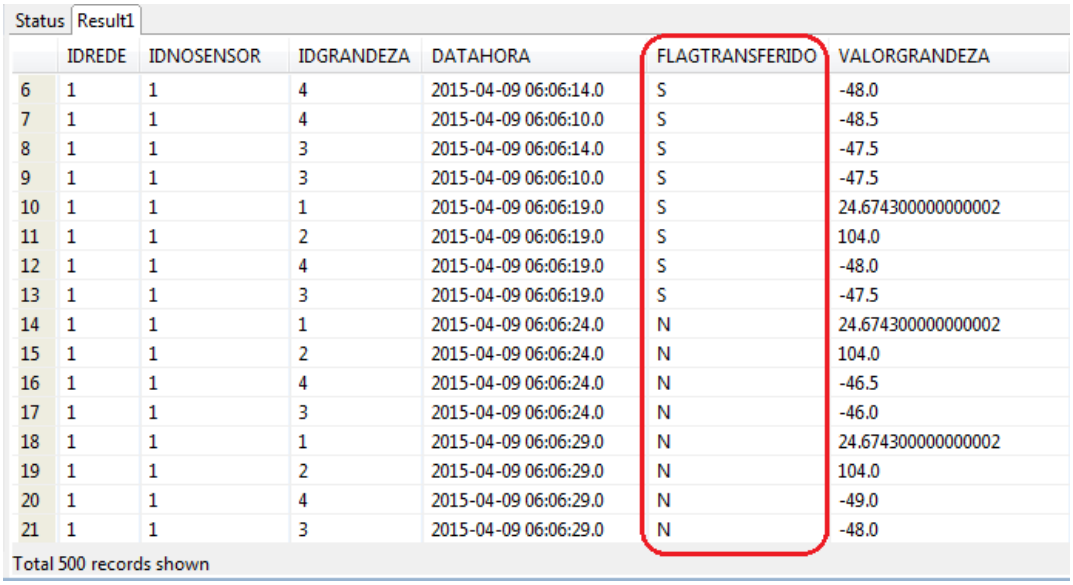
As informações são enviadas do *Publisher* HTTP para o SMR através de chamadas à *web services*. Nesse caso, o SMR é o servidor e o *Publisher* HTTP, através do MP, faz o papel de cliente de tais *web services*.

As solicitações de envio de informações ocorrem dentro de uma peridiocidade especificada no Cadastro da RSSF, mais especificamente pelo campo Período de Leitura, que deve ser informado em milisegundos.

Conforme as informações são enviadas para o SMR, o MP atualiza os registros do banco de dados, alterando o valor do campo *FlagTransferido*, de N (Não) para S (Sim). Dessa forma evita-se que um mesmo registro seja enviado mais de uma vez.

A Figura 21 mostra as duas situações. Um conjunto de registros de monitoramento que já foram enviados ao SMR, contendo o valor "S" no campo *FlagTransferido*, e outro conjunto, contendo o valor "N", representando registros que ainda não haviam sido enviados.

Figura 21: Registros de monitoramento enviados e não-enviados ao SMR.



Status | Result1

	IDREDE	IDNOSENSOR	IDGRANDEZA	DATAHORA	FLAGTRANSFERIDO	VALORGRANDEZA
6	1	1	4	2015-04-09 06:06:14.0	S	-48.0
7	1	1	4	2015-04-09 06:06:10.0	S	-48.5
8	1	1	3	2015-04-09 06:06:14.0	S	-47.5
9	1	1	3	2015-04-09 06:06:10.0	S	-47.5
10	1	1	1	2015-04-09 06:06:19.0	S	24.674300000000002
11	1	1	2	2015-04-09 06:06:19.0	S	104.0
12	1	1	4	2015-04-09 06:06:19.0	S	-48.0
13	1	1	3	2015-04-09 06:06:19.0	S	-47.5
14	1	1	1	2015-04-09 06:06:24.0	N	24.674300000000002
15	1	1	2	2015-04-09 06:06:24.0	N	104.0
16	1	1	4	2015-04-09 06:06:24.0	N	-46.5
17	1	1	3	2015-04-09 06:06:24.0	N	-46.0
18	1	1	1	2015-04-09 06:06:29.0	N	24.674300000000002
19	1	1	2	2015-04-09 06:06:29.0	N	104.0
20	1	1	4	2015-04-09 06:06:29.0	N	-49.0
21	1	1	3	2015-04-09 06:06:29.0	N	-48.0

Total 500 records shown

Fonte: Elaboração própria.

Outra funcionalidade do MP, é a remoção dos registros de monitoramento já enviados ao SMR, do banco de dados. Isso evita que o banco de dados do *Publisher* HTTP fique sobrecarregado com informações já disponíveis para consulta na Internet, e permite que a sua performance mantenha-se elevada.

Em termos práticos, o MP realiza pesquisas no banco de dados, em cada uma das tabelas, buscando por registros que possuem o valor "N" no campo *FlagTransferido*. Caso as pesquisas retornem algum registro, ele será enviado ao SMR através de uma chamada ao *web service* desenvolvido para receber aquela informação específica. Por exemplo, para enviar informações referentes às grandezas monitoradas, o MP realiza uma pesquisa na tabela Monitoramento, e caso encontre algum registro para ser enviado ao SMR, ele fará uma chamada ao *web service RecebeDadosMonitoramentoWs*. Tais *web services* serão apresentados mais detalhadamente no próximo tópico desse capítulo.

O *Publisher* HTTP define um único período de execução dessa funcionalidade em vinte e quatro horas. Dessa forma, deve-se informar no campo "Horário para remoção de dados" do cadastro da RSSF, o horário em que a transferência ocorrerá diariamente.

6.2.2. O Sistema de Monitoramento Remoto (SMR)

O SMR também foi desenvolvido como uma aplicação web utilizando a linguagem Java, assim como o SML.

Esse sistema integra efetivamente o *Publisher* HTTP à internet, uma vez que ele está preparado para receber as informações enviadas e disponibilizá-las, tanto para consultas diretas através de sua interface acessada por navegadores web, quanto através de sua interface baseada em *web services*. Esta última, permite a integração com sistemas de terceiros que também sejam compatíveis com tal tecnologia. Além disso, é por meio dessa interface que as informações provenientes da RSSF poderão ser disponibilizadas para aplicações móveis utilizadas em *smartphones* e *tablets*.

O envio das informações provenientes da RSSF, do *Publisher* HTTP para o SMR, também é realizado por meio da interface de *web services*.

O SMR está dividido em 2 módulos:

- Módulo de Monitoramento Remoto (MMR)
- Módulo *Web Services* (MWS)

As funções de monitoramento disponibilizadas pelo MMR estão relacionadas à interface com o usuário e são praticamente as mesmas encontradas no componente MGL do *Publisher* HTTP, mas que, no entanto, podem ser acessadas via Internet. Tais funcionalidades incluem a geração de listagens dos componentes do sistema, relatórios e gráficos criados de acordo com os dados captados pela RSSF.

É importante ressaltar que as funções de cadastro, como inclusão, alteração e exclusão de dados, não estão disponíveis no MMR. Ou seja, o usuário só terá acesso à funcionalidades de consulta através dessa interface.

Uma das responsabilidades do Módulo *Web Services* (MWS) é receber as informações provenientes da RSSF enviadas pelo *Publisher* HTTP, e persisti-las no banco de dados local utilizado pelo SMR. Nesse caso, o SMR atua como um servidor, disponibilizando *web services* que tem a função de receber dados

enviados pelo *Publisher* HTTP, o qual, no caso, atua como um cliente desses web services. O módulo de software do *Publisher* HTTP que efetivamente faz o envio das informações para o MWS, é o Módulo Publicador (MP).

O MWS também é responsável por disponibilizar os dados utilizados pelas funcionalidades de monitoramento do MMR por meio de *web services*. Isto permite que o SMR seja integrado à outros sistemas de terceiros que utilizam essa mesma tecnologia, e que novos sistemas de monitoramento sejam desenvolvidos com base nos *web services* disponibilizados pelo MWS.

Dessa forma, pode-se definir o MWS como um conjunto de *web services*, cada qual executando uma função bastante específica. Esse conjunto de *web services* pode ser dividido em dois grupos. Um deles destinado ao recebimento de informações enviadas por clientes desses *web services*, e o outro, relacionado ao fornecimento de informações solicitadas por tais clientes.

Os *web services* do MWS foram desenvolvidos com a linguagem Java, seguindo o padrão REST, pois, diferentemente do padrão SOAP, que é executado sobre o protocolo HTTP, o REST utiliza o próprio HTTP na troca de informações entre as partes cliente e servidor.

Em relação ao recebimento de informações provenientes do *Publisher* HTTP, o MWS define *web services* tanto para receber informações referentes à configuração e funcionamento da RSSF, quanto para receber dados captados pela rede de sensores e os valores calculados a partir deles. A Tabela 8, exibe todos os *web services* criados com tais finalidades.

Tabela 8: *Web services* para recebimento de dados.

Web service	Funcionalidade do web service
RecebeDadosRedeWs	Recebe os dados de configuração da RSSF e os armazena na tabela Rede.
RecebeDadosSensorWs	Recebe os dados dos sensores cadastrados e os armazena na tabela Sensor.
RecebeDadosGrandezaWs	Recebe os dados das grandezas monitoradas e os armazena na tabela Grandeza.
RecebeDadosGrandezaDataPointWs	Recebe os dados dos bytes do pacote Radiuino que correspondem a cada uma das grandezas monitoradas. Essas Dados são armazenados na tabela GrandezaDataPoint.
RecebeDadosMonitoramentoWs	Recebe os dados das grandezas calculadas e os armazena na tabela Monitoramento.

Fonte: Elaboração própria.

A Figura 22 exibe o código-fonte escrito em Java, utilizado para criar o *web service* responsável pelo recebimento das informações de monitoramento.

Figura 22: Código-fonte Java do *web service* MonitoramentoWs.

Fonte: Elaboração própria.

Os demais *web services* que compõem o MWS, foram criados para disponibilizar as informações armazenadas no banco de dados do SMR através dessa tecnologia. Essas informações são as mesmas utilizadas pelo MMR, por exemplo, para gerar relatórios e gráficos. No entanto, elas podem ser obtidas por

sistemas de terceiros e aplicativos móveis, podendo ser utilizadas em novas funcionalidades, de acordo com a necessidade de cada aplicação.

A Tabela 9 exibe a relação dos *web services* do MWS voltados para a disponibilização das informações armazenadas pelo sistema, incluindo tanto as informações de configuração da RSSF, quanto aquelas utilizadas para efetuar o monitoramento das grandezas em questão.

Tabela 9: *Web services* para disponibilização de dados.

Web service	Funcionalidade do web service
ForneceDadosRedeWs	Fornece os dados de configuração da RSSF armazenados na tabela Rede.
ForneceDadosSensorWs	Fornece os dados dos sensores cadastrados armazenados na tabela Sensor.
ForneceDadosGrandezaWs	Fornece os dados das grandezas monitoradas armazenados na tabela Grandeza.
ForneceDadosGrandezaDataPointWs	Fornece os dados dos bytes do pacote Radiuino que correspondem a cada uma das grandezas monitoradas, provenientes da tabela GrandezaDataPoint.
ForneceDadosMonitoramentoWs	Fornece os dados das grandezas calculadas, armazenados na tabela Monitoramento.

Fonte: Elaboração própria.

Na Figura 23, observa-se o conteúdo recebido do *web service ForneceDadosRedeWs*, no formato XML, quando ocorre uma solicitação por algum cliente desse *web service*.

Figura 23: Arquivo XML recebido do *web service* ForneceDadosRedeWs.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rede>
  <baudrate>9600</baudrate>
  <databits>5</databits>
  <habilitarLeitura>SIM</habilitarLeitura>
  <habilitarRemocao>SIM</habilitarRemocao>
  <habilitarTransferencia>SIM</habilitarTransferencia>
  <horarioRemocao>21:00:00</horarioRemocao>
  <idNoBase>0</idNoBase>
  <idRede>1</idRede>
  <nomeRede>Rede 1 - Radiuino</nomeRede>
  <parity>0</parity>
  <periodoLeitura>5000</periodoLeitura>
  <periodoTransferencia>60000</periodoTransferencia>
  <portaCOM>/dev/USB0</portaCOM>
  <stopbits>1</stopbits>
  <timeout>2000</timeout>
</rede>
```

Fonte: Elaboração própria.

O banco de dados utilizado para o SMR, possui a mesma estrutura do banco de dados definido para o *Publisher* HTTP.

6.2.3. O servidor de banco de dados e o container *web*

Deve ser lembrado que essa aplicação web necessita de um servidor de aplicações ou container web para ser executada, e de um servidor de banco de dados para armazenar tanto as suas informações de configuração quanto aquelas que serão monitoradas e controladas pelo sistema.

Para a definição de ambos, deve ser levado em consideração as características do hardware definido para o Publisher HTTP, que, por questões de baixo custo e tamanho físico reduzido, possui recursos limitados.

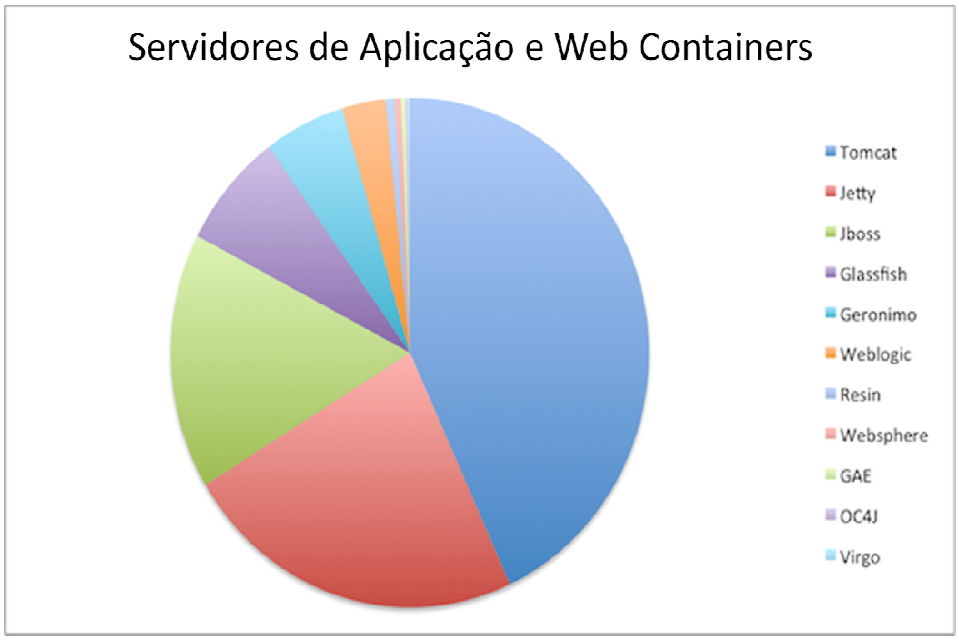
Baseado nessa premissa, buscou-se opções de software capazes de serem executadas em um hardware com essas características, que apresentassem um desempenho razoável e que consumissem o mínimo de recursos, principalmente em relação à memória. Além disso, levou-se em conta, apesar das soluções avaliadas serem open-source, questões como o grau de consolidação das soluções no mercado e o suporte por parte do fornecedor e das comunidades que contribuem para o desenvolvimento.

Em relação ao servidor de aplicações ou container *web*, entendeu-se que o porte da aplicação a ser desenvolvida para o *Publisher* HTTP era pequeno, portanto, não seria necessária a utilização de um servidor de aplicações, um container web seria suficiente. Para manter uma compatibilidade entre o container web utilizado no Publisher HTTP e aquele no qual o SMR seria instalado, no ambiente de computação em nuvem da empresa Amazon Web Services - AWS (AWS, 2015), optou-se pela utilização do Apache Tomcat (APACHE TOMCAT, 2015), suportado pela Apache Software Foundation (APACHE, 2015). Pois, este é o único container *web*, de menor porte, disponibilizado por esta empresa. É importante ressaltar que o Apache Tomcat atende à todos os pré-requisitos citados anteriormente.

De forma complementar, para demonstrar que o Apache Tomcat, apesar de ser a única opção da AWS, apresenta uma superioridade em relação às demais opções do mercado, principalmente em termos de utilização de recursos, foi realizada uma comparação com o container *web* Jetty, suportado pelo projeto Eclipse (ECLIPSE, 2015).

Baseado em dados divulgados pela empresa Plumbr (PLUMBR - POPULARES, 2013), especializada em análises e soluções de desempenho para aplicações desenvolvidas para a plataforma Java, sobre os containeres *web* mais utilizados atualmente, o Tomcat apresentou uma porcentagem de utilização de 43% e o Jetty 23%, como mostra a Figura 24.

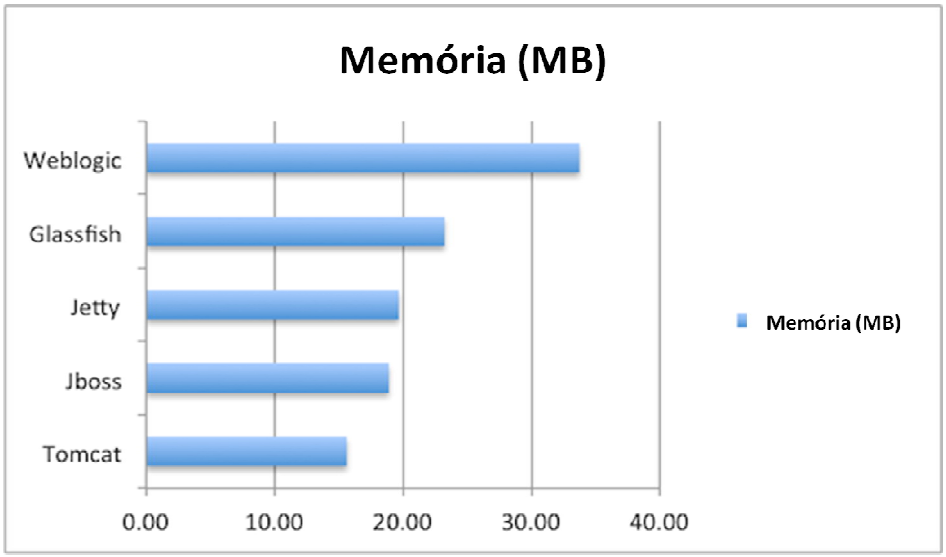
Figura 24: Servidores de aplicação e containeres *web* mais populares.



Fonte: (PLUMBR - POPULARES, 2013).

Em relação ao consumo de memória RAM apresentado por ambos (PLUMBR - PERFORMANCE, 2013), o Tomcat consome cerca de 15,63MB contra 19,66MB do Jetty, conforme mostra a Figura 25. Uma consideração importante devido a limitação dos recursos de *hardware* do *Publisher* HTTP.

Figura 25: Consumo de memória do Tomcat e do Jetty.



Fonte: (PLUMBR - PERFORMANCE, 2012).

Em relação ao servidor de banco de dados, os pré-requisitos definidos para a seleção é que consumisse poucos recursos de *hardware*, principalmente memória, que fosse capaz de rodar embutido na própria aplicação, buscando eliminar a necessidade de instalação e um melhor desempenho por parte da UI. Por fim, havia a necessidade do servidor ser compatível com a tecnologia Java, uma vez que ele deveria rodar a partir de dentro da aplicação *web*.

A definição partiu de um estudo comparativo realizado pela empresa responsável pelo desenvolvimento do servidor de banco de dados H2 (H2, 2015), comparando o próprio H2 com outras duas opções também muito utilizados, o HSQLDB (HSQLDB, 2015) e o Apache Derby (APACHE DERBY, 2015). Todos os três servidores de banco de dados são desenvolvidos com a tecnologia Java e podem rodar embutidos em aplicativos.

Na comparação entre os servidores foram realizados testes de inicialização, execução de consultas e de operações de inserção, atualização e exclusão de dados, levando-se em consideração o tempo para a realização de cada operação e o quanto cada uma delas consumiu da memória principal do hardware utilizado.

A Tabela 10, mostrada a seguir, foi criada com base nos resultados desse estudo (H2 DATABASE PERFORMANCE, 2014), considerando somente o quesito consumo de memória, destacando o consumo máximo, mínimo e médio de memória para cada servidor de banco de dados.

Tabela 10: Comparação entre os servidores H2, HSQLDB e Apache Derby.

Servidor de Banco de dados	Consumo Máximo (MB)	Consumo Mínimo (MB)	Consumo Médio (MB)
H2	21	12	16,5
HSQLDB	22	10	16
Apache Derby	9	7	8

Fonte: Elaboração própria.

6.3. Metodologia dos Testes e Resultados

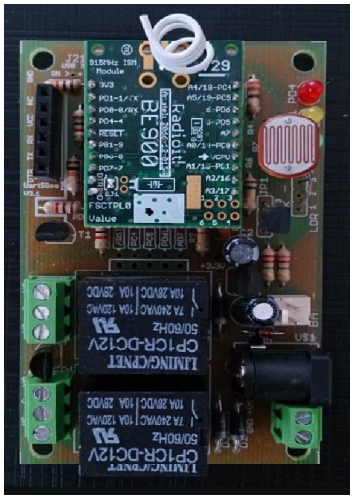
A metodologia dos testes realizados teve por objetivo buscar evidências que comprovassem a validade da proposta, demonstrando através de uma abordagem essencialmente prática, a sua aplicabilidade em sistemas reais.

Todos os experimentos foram realizados em ambiente de laboratório utilizando a mesma RSSF, implementada através da plataforma Radiuino, por se tratar de uma plataforma baseada em *hardware* e *software* de código-aberto, com um ambiente de desenvolvimento intuitivo e de fácil utilização. Outros fatores que levaram a escolha da plataforma Radiuino são:

- Implementação de pequenas redes de sensores de forma simples;
- Arquitetura voltada a para criação de redes fim a fim;
- Facilidade de aquisição e baixo custo do hardware.

Os nós-sensores utilizaram uma placa de aplicação com dois relés, um transdutor de temperatura, um resistor sensível a espectro luminoso, diodos emissores de luz e um rádio BE900, que consiste em um módulo Arduino com um transceptor de rádio freqüência integrado, capaz transmitir e receber dados sem fio na faixa de 915MHz, como podemos observar na Figura 26.

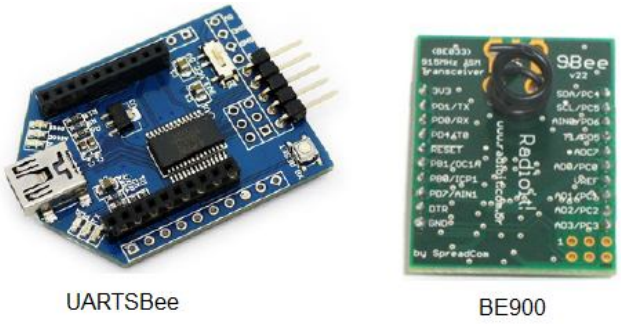
Figura 26: Placa de aplicação Radiuino.



Fonte: Elaboração própria.

Para o nó-base, foi utilizado o rádio BE900 acoplado a um gravador UartSBee. O módulo BE900, além do chip CC1101 que é responsável pela interface de rádio, também é composto pelo micro-controlador ATMega 368 que realiza o processamento de informações da rede e o controle do nó. A Figura 27 exibe um gravador UARTSBee e um módulo BE900.

Figura 27: gravador UARTSBee e um módulo BE900.



Fonte: Modificado de (RADIUINO, 2014).

Para a programação e gravação do *firmware* nos nós base e sensor, foram utilizadas a IDE do projeto Arduino e o gravador UartSBee.

O Publisher HTTP utilizou como *hardware* o micro-controlador Beaglebone Black para todos os testes.

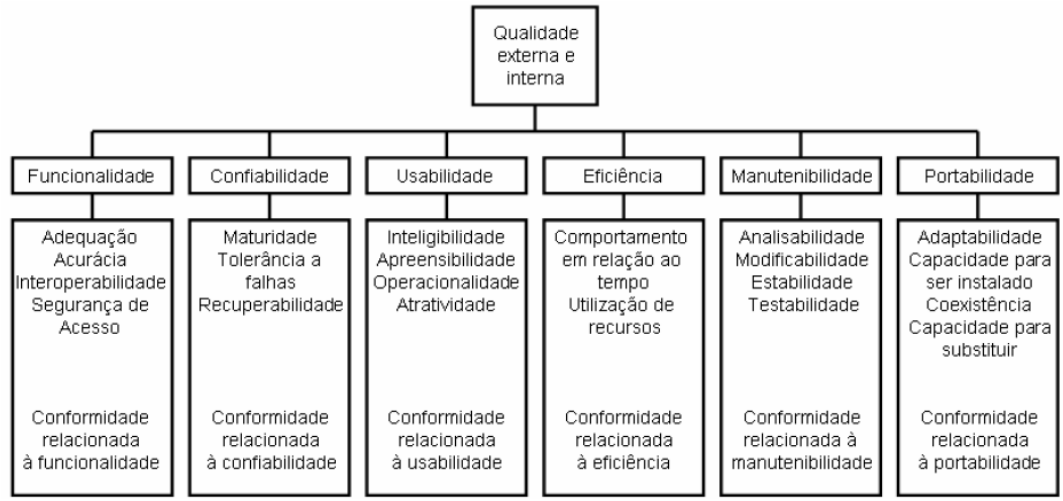
Em todos os cenários de teste, a comunicação física do *Publisher* HTTP com o nó-base, se deu através de uma saída USB localizada no micro-controlador Beaglebone Black, onde o gravador UartSBee dotado de um rádio BE900, foi acoplado.

Como a proposta apresentada por este trabalho se trata de uma solução de integração entre redes de diferentes padrões baseada em *software*, optou-se pela utilização da norma NBR ISO/IEC 9126 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2003) para a definição dos itens de qualidade de *software* a serem avaliados, através de testes baseados em técnicas de caixa-preta sugeridos pelo *Guide to the Software Engineering Body of Knowledge* (IEEE COMPUTER SOCIETY, 2014) do IEEE *Computer Society*. Especificamente para os testes

realizados com *web services* foi utilizado de forma complementar o OASIS *Web Services Quality Model* (ADVANCING OPEN STANDARDS FOR THE INFORMATION SOCIETY, 2011).

O NBR ISO/IEC 9126 é uma norma da ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT) baseada no conjunto de normas ISO/IEC 9126, que estabelece um modelo de qualidade para produto de software, avaliando tanto requisitos de qualidade interna quanto externa, conforme mostra a Figura 28.

Figura 28: Modelo de qualidade externa e interna da NBR ISO/IEC 9126.



Fonte: Elaboração própria.

A qualidade interna é medida e avaliada com relação aos requisitos de implementação do *software*, ou seja, referem-se a métricas de um produto de software a partir de suas características internas, sem a necessidade de execução.

Já a qualidade externa é medida e avaliada com base em métricas obtidas quando o *software* é executado, normalmente num ambiente de testes.

Com base no modelo da NBR ISO/IEC 9126 e no *Guide to the Software Engineering Body of Knowledge* (IEEE COMPUTER SOCIETY, 2014), definiu-se os seguintes itens do modelo para compor os testes que foram executados:

Funcionalidade: avaliação da acuracidade ou precisão do sistema em relação a quantidade de informação lida e armazenada localmente no *Publisher* HTTP, e a quantidade enviada e armazenada no SMR. A qualidade das informações também

será avaliada confrontando o conteúdo das informações armazenadas no *Publisher* HTTP e no SMR para verificação da integridade das informações;

Confiabilidade: a tolerância a falhas do sistema foi avaliada. Uma vez que o sistema depende de uma conexão com a Internet para que os dados sejam publicados no SMR, avaliou-se o seu comportamento e o grau de comprometimento que uma possível falta de conexão causaria ao sistema;

Eficiência: nesse item foi avaliado o tempo de resposta do web services que suporta a integração entre as duas redes.

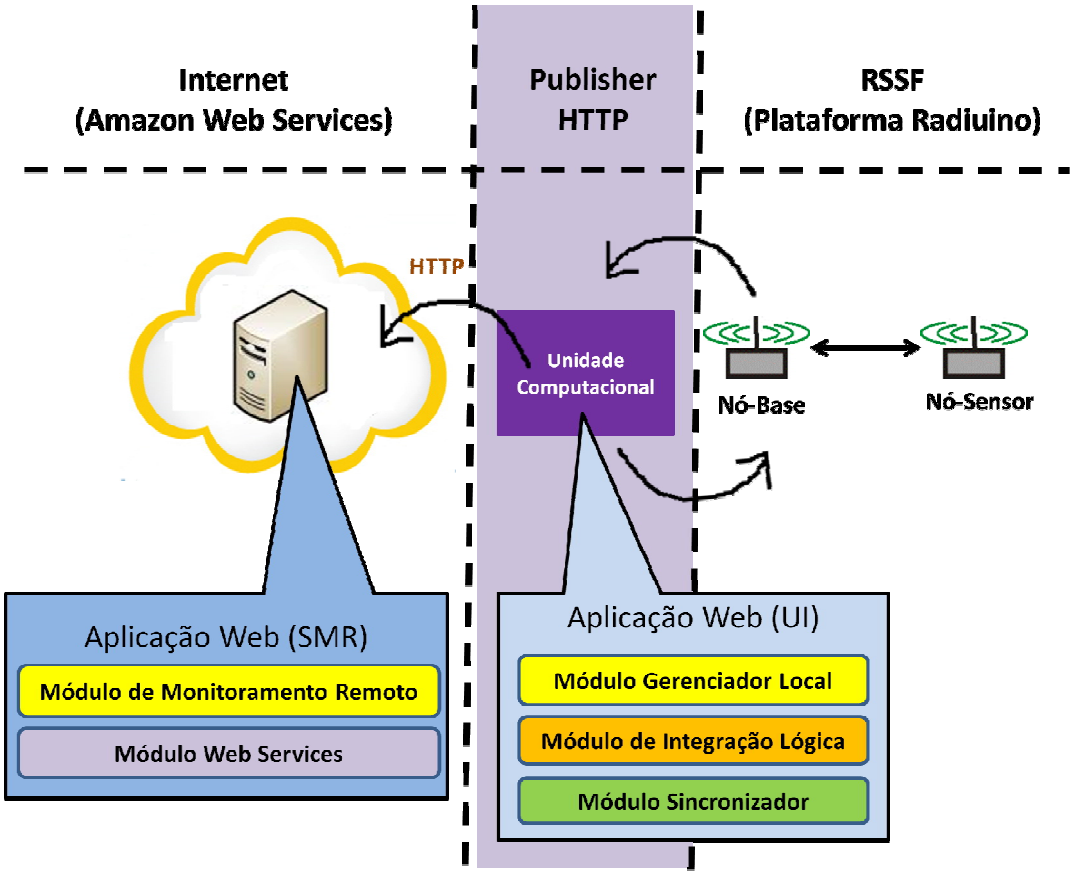
6.3.1. Cenário 1: Testando a Funcionalidade do Sistema

O primeiro cenário de testes tem por finalidade validar a funcionalidade de um sistema baseado no *Publisher* HTTP, comprovando a eficácia da proposta. Ou seja, a partir de uma solicitação de leitura de uma ou mais grandezas, feita pelo *Publisher* HTTP, a RSSF deve retornar um pacote ao solicitante contendo tais valores. O *Publisher* HTTP, ao receber o pacote da RSSF obtém os valores desejados, efetua os cálculos necessários de acordo com cada grandeza monitorada e armazena essas informações no seu banco de dados local. Desse ponto em diante, numa situação de sucesso, o monitoramento já poderá ser realizado por qualquer usuário que esteja na mesma intranet do *Publisher* HTTP, com a utilização de um navegador *web* acessando a interface onde relatórios e gráficos são gerados.

O ciclo de processamento do sistema se torna completo quando os dados armazenados no banco de dados local do *Publisher* HTTP são enviados, com a utilização de chamadas à web services, para o SMR, localizado na Internet.

A Figura 29 exibe uma visão geral sobre o sistema implementado para este experimento.

Figura 29: Visão geral do sistema montado para o primeiro experimento.

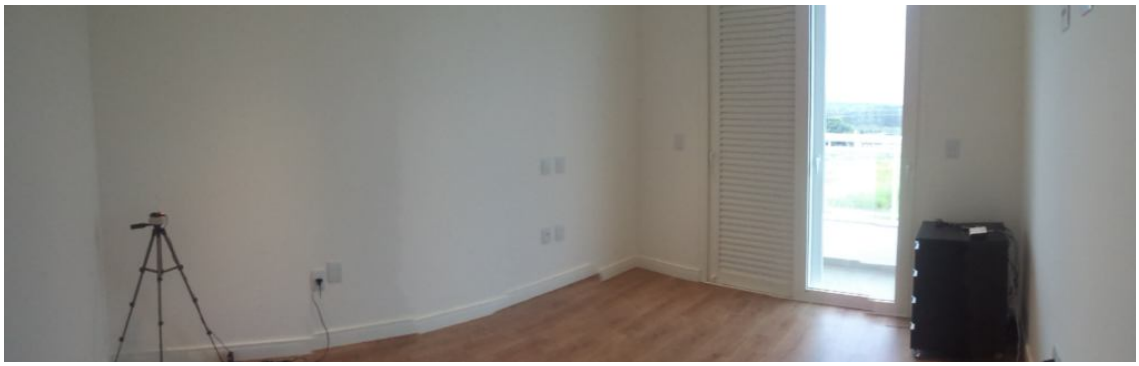


Fonte: Elaboração própria.

Para esse experimento foi utilizada uma RSSF do tipo ponto a ponto, composta por um único nó-sensor e um nó-base, o qual foi conectado a uma interface USB do *hardware* utilizado pelo *Publisher* HTTP. O *Publisher* HTTP foi conectado por meio de uma interface RJ-45 a um modem Datacom modelo DM2270, que conectava o *Publisher* HTTP a uma rede local de 100 Mbps, acessando a Internet por uma conexão ADSL que apresentava, no momento do teste, 700 Kbps de *Upstream* e 2.303 Mbps de *Downstream*.

A RSSF foi implementada em uma sala fechada, escolha que se deu pelo foco do trabalho estar voltado para ambientes fechados como estufas para cultivo de frutas e hortaliças. As medidas da sala são 2,90 metros de largura, 4,30 metros de comprimento e 3 metros de altura. A Figura 30 exibe uma visão panorâmica da sala.

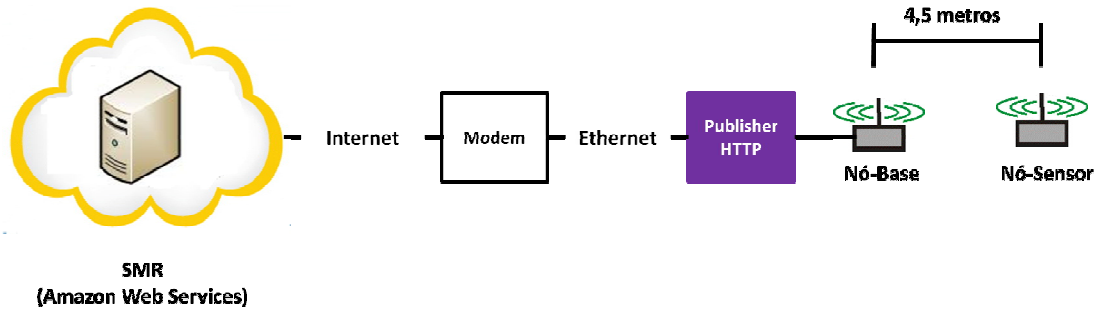
Figura 30: Foto panorâmica da sala utilizada para realização do experimento.



Fonte: Elaboração própria.

O nó-base e o nó-sensor foram posicionados, respectivamente, a uma altura de 70 cm e 90 cm do chão, numa disposição diagonal. A distância existente entre o nó-base e o nó-sensor era de 4,5 metros, conforme pode-se observar no diagrama exibido pela Figura 31.

Figura 31: Disposição física dos elementos utilizados no experimento.



Fonte: Elaboração própria.

A RSSF foi definida para atuar com uma potência de 10 mW. Para o canal de comunicação, foi adotado o valor zero, padrão definido nos *firmwares* Radiuino, uma vez que não havia outras RSSFs instaladas no local do teste e não havia risco de interferências de outros dispositivos. A freqüência de modulação foi definida em 915 MHz. Foi utilizado *pooling* como técnica de acesso ao meio na RSSF.

O *software* da UI foi instalado em um mini-computador Beaglebone Black, utilizando sistema operacional Linux Ubuntu especifico para micro-controladores

ARM. O container *web* e o banco de dados relacional utilizados foram respectivamente o Apache Tomcat versão 7 e o Apache Derby 10.11.

Para instalação do SMR, foi criado um *host* virtual disponível na *Internet*, dentro do ambiente de computação em nuvem disponibilizado pela empresa Amazon Web Services (*AWS*). O sistema operacional utilizado foi o Ubuntu Linux Server 14.04, e seguindo as definições realizadas para o Publisher HTTP, foram utilizados o Apache Tomcat versão 7 e o Apache Derby 10.11.

Ambos os ambientes utilizam a versão 7 da Java Virtual Machine.

O *Publisher* HTTP foi configurado para solicitar dados à RSSF a cada 30 segundos, enviando os dados para o host na internet a cada 45 segundos.

O item de qualidade funcionalidade, da NBR ISO/IEC 9126, foi avaliado com base na acuracidade ou precisão das informações manipuladas pelo *Publisher* HTTP.

6.3.1.1. Avaliando a Precisão de Forma Quantitativa

Segundo a NBR ISO/IEC 9126 esse item deve ser avaliado de forma a se comprovar que um determinado conjunto de dados especificado como resultado para uma determinada funcionalidade do sistema, está compatível com o resultado real.

Inicialmente, buscou-se medir a quantidade de registros armazenados no banco de dados local do *Publisher* HTTP dentro de um determinado período de tempo, e comparar com a quantidade de registros enviada e armazenada no banco de dados do SMR, dentro do mesmo período. Foram definidos períodos de avaliação de 12, 24 e 36 horas de operação do sistema.

Cada solicitação de leitura feita à RSSF gera 4 registros na tabela Monitoramento, ou seja, um para cada uma das grandezas monitoradas: temperatura, luminosidade, RSSI Down e RSSI Up.

Como o sistema foi configurado para solicitar dados à RSSF de 30 em 30 segundos, os períodos de avaliação definidos devem apresentar aproximadamente

as quantidades de registros nos bancos de dados do *Publisher* HTTP e do SMR exibidos pela Tabela 11:

Tabela 11: Períodos de avaliação e quantidades de registros previstas.

Período de avaliação (horas)	Quantidade de registros prevista
12	5.760
24	11.520
36	17.280

Fonte: Elaboração própria.

O processo de envio das informações do *Publisher* HTTP para o SMR, é uma parte do sistema que depende de serviços fornecidos por terceiros, como é o caso do acesso à Internet. Prevendo esse tipo de problema, o sistema foi implementado com um mecanismo de recuperação a falhas que o impeçam de enviar os dados para o SMR. Dessa forma o sistema pode operar normalmente solicitando informações à RSSF e enviá-las ao SMR quando a conexão for restabelecida.

Para que fique explícita essa dependência, e que a falha em tais serviços leva à falha do sistema, este teste foi dividido em duas partes. Na primeira, serão exibidos os resultados de funcionamento do sistema sem a utilização do mecanismo de recuperação a falhas, para que se possa medir tais ocorrências. Na segunda parte, os resultados são de uma execução do sistema por completo, incluindo a atuação do mecanismo de recuperação a falhas.

As consultas para obtenção dos resultados foram executadas diretamente no banco de dados de cada aplicação com a utilização Structured *Query Language* (SQL) e o *software* Eclipse, que atuou como uma interface cliente, e ao qual os bancos de dados foram conectados para a realização das consultas. O campo utilizado como base para a consulta foi a data de solicitação à RSSF.

A Tabela 12 exibe os comandos SQL utilizados para cada um dos períodos de avaliação definidos:

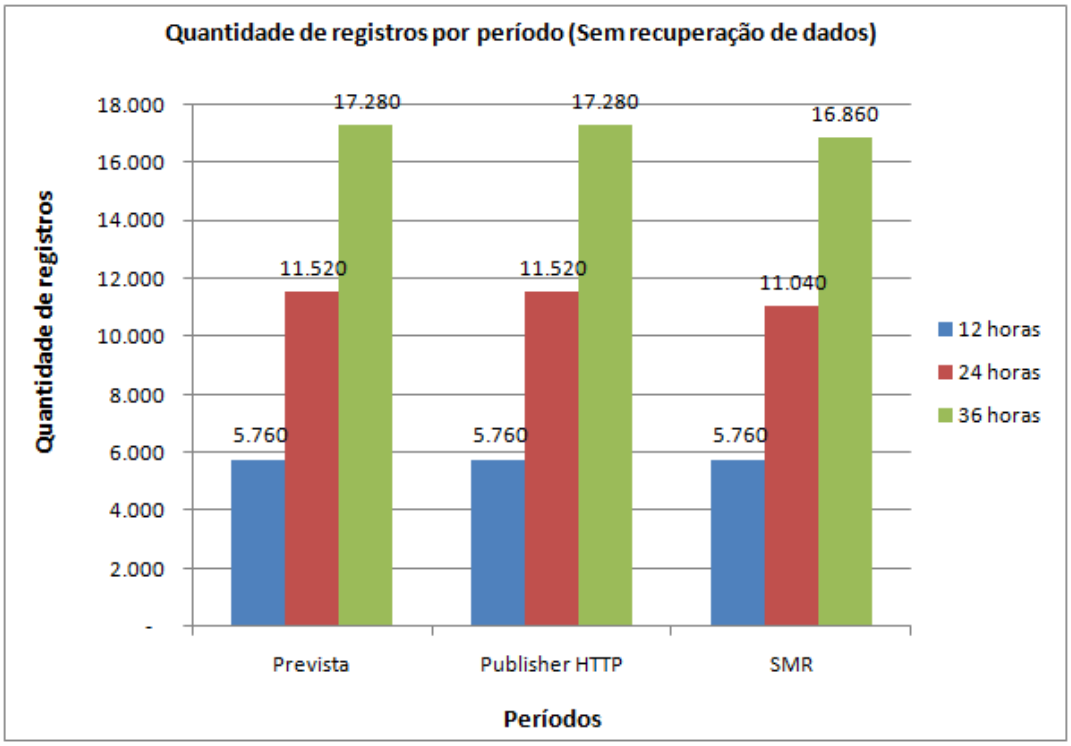
Tabela 12: Comandos SQL utilizados nas consultas sem recuperação de dados.

Período de avaliação (horas)	Comando SQL
12	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-02-25 12:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-02-25 23:59:59.0'
24	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-03-17 00:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-03-17 23:59:59.0'
36	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-04-02 00:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-04-03 12:00:00.0'

Fonte: Elaboração própria.

Esses resultados foram obtidos sem a utilização do mecanismo de recuperação a falhas para envio dos dados ao SMR. Nesse caso, foi identificada uma diferença na quantidade de registros do SMR em relação à quantidade armazenada no *Publisher* HTTP, e conseqüentemente, ao que havia sido previsto, conforme podemos observar na Figura 32. O período de medição referente a 12 horas não apresentou diferença entre o que foi previsto e as quantidades identificadas nos bancos de dados de ambas as aplicações.

Figura 32: Gráfico de quantidade de registros por período avaliado sem recuperação de dados.



Fonte: Elaboração própria.

Essa diferença se justifica pelas falhas de transferência de dados que foram identificadas através dos registros de atividade do *Publisher* HTTP. A Tabela 13 exibe o tempo de duração das falhas identificadas e a quantidade de registros acumulada no banco de dados do *Publisher* HTTP durante a falta de conexão com a Internet.

Tabela 13: Falhas de envio de dados do *Publisher* HTTP para o SMR.

Data	Duração da falta de conexão (hh:mm:ss)	Quantidade de registros acumulada
17/03/2015	01:00:22	480
02/04/2015	00:37:15	296
03/04/2015	00:15:45	124

Fonte: Elaboração própria.

Para verificar a quantidade de registros considerando a utilização do recurso de recuperação a falhas no envio dos dados para o SMR, foram definidos novos

períodos de avaliação. Os comandos SQL exibidos na Tabela 14 referem-se a tais definições.

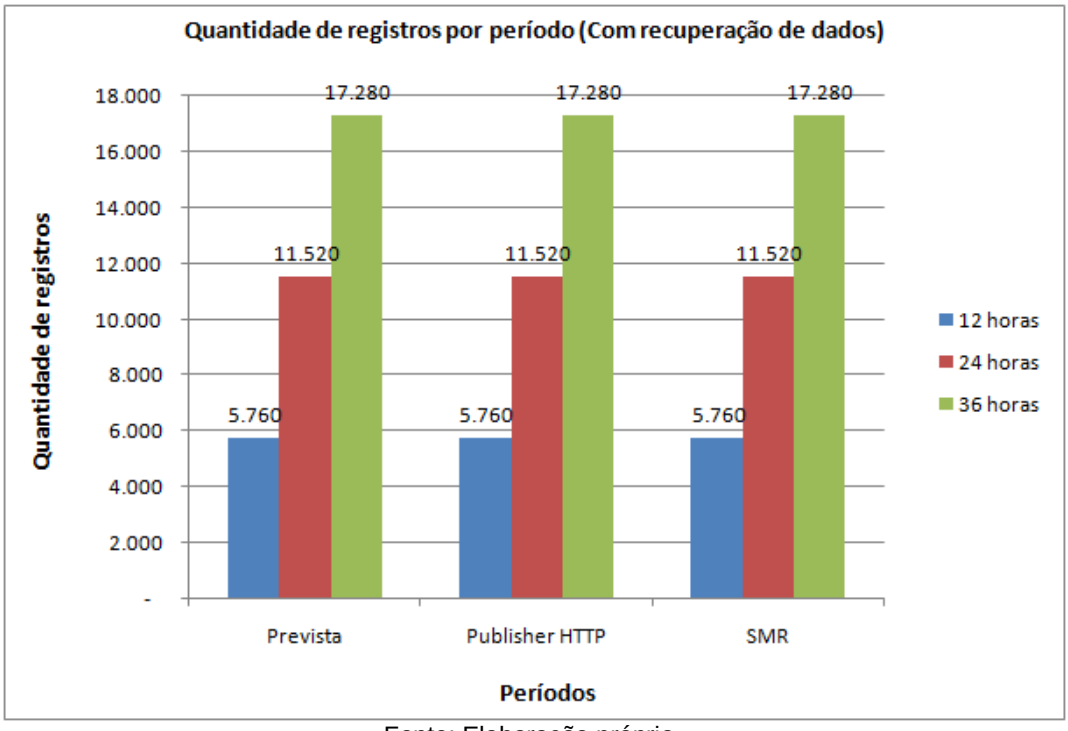
Tabela 14: Comandos SQL utilizados nas consultas com recuperação de dados.

Período de avaliação (horas)	Comando SQL
12	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-02-27 12:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-02-27 23:59:59.0'
24	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-03-19 00:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-03-19 23:59:59.0'
36	select count(*) from MONITORAMENTO where date(DATAHORASOLICITACAORSSF) >= '2015-04-05 00:00:00.0' and date(DATAHORASOLICITACAORSSF) <= '2015-04-06 12:00:00.0'

Fonte: Elaboração própria.

Os resultados desse teste apresentaram, não só uma quantidade de registros equivalente ao que foi previsto, mas também igual entre os bancos de dados do *Publisher* HTTP e do SMR, conforme podemos observar na Figura 33.

Figura 33: Gráfico de quantidade de registros por período avaliado com recuperação de dados.



Fonte: Elaboração própria.

É importante ressaltar que existiram falhas no processo de envio dos dados do *Publisher* HTTP para o SMR, devido a perda de conexão com a Internet, uma vez que esse é um serviço fornecido por um terceiro. O fato de ambos os servidores apresentarem quantidades de registros iguais para o teste em questão, se justifica pela presença de mecanismos de tolerância a falhas e recuperabilidade, presentes no sistema, e que permitiram que os dados fossem transferidos ao SMR quando a conexão com a Internet foi restabelecida. Esses mecanismos serão explicados detalhadamente no próximo item de qualidade a ser analisado, a confiabilidade do sistema.

O processo de solicitação de dados à RSSF e armazenamento dessas informações no banco de dados local do *Publisher* HTTP, não apresentou falhas.

Dessa forma, pode-se então concluir que em termos quantitativos o sistema estava operando corretamente.

6.3.1.2. Avaliando a Precisão de Forma Qualitativa

O próximo teste executado relacionado à precisão, avaliou o sistema qualitativamente, de forma a identificar se as informações armazenadas no banco de dados local do *Publisher* HTTP eram as mesmas transferidas e armazenadas no banco de dados do SMR, e se os dados estavam íntegros. O intuito foi avaliar se não houve corrupção dos dados no processo de transferência executado do *Publisher* HTTP para o SMR, e no armazenamento dessas informações.

Para obtenção das informações a partir dos bancos de dados do *Publisher* HTTP e do SMR, foi utilizada novamente a linguagem SQL considerando os dados obtidos a partir da RSSF e transferidos ao SMR na data do dia 06/04/2015. O código de identificação de cada grandeza monitorada cadastrada no sistema, também foi utilizado na filtragem dos dados, para que cada uma delas fosse analisada separadamente.

A Tabela 15 exibe os códigos de identificação de cada uma das grandezas cadastradas no sistema para que se possa compreender os comandos SQL que serão exibidos mais adiante.

Tabela 15: Códigos de identificação das grandezas monitoradas.

Código de Identificação	Grandeza
1	Temperatura
2	Luminosidade
3	RSSI Up
4	RSSI Down

Fonte: Elaboração própria.

A partir dos comandos SQL exibidos na Tabela 16, foram obtidos os valores usados para se calcular a média aritmética e o desvio padrão para cada uma das grandezas monitoradas.

Tabela 16: Comandos SQL utilizados para as consultas qualitativas

Grandeza	Comando SQL
Temperatura	select valorgrandeza from MONITORAMENTO where (IDGRANDEZA = 1) and (date(DATAHORASOLICITACAORSSF) = '2015-04-06')
Luminosidade	select valorgrandeza from MONITORAMENTO where (IDGRANDEZA = 2) and (date(DATAHORASOLICITACAORSSF) = '2015-04-06')
RSSI Up	select valorgrandeza from MONITORAMENTO where (IDGRANDEZA = 3) and (date(DATAHORASOLICITACAORSSF) = '2015-04-06')
RSSI Down	select valorgrandeza from MONITORAMENTO where (IDGRANDEZA = 4) and (date(DATAHORASOLICITACAORSSF) = '2015-04-06')

Fonte: Elaboração própria.

Esses comandos foram executados tanto no banco de dados do *Publisher* HTTP quanto do SMR, apresentando exatamente os mesmos resultados. Portanto, tais informações serão exibidas numa única tabela representando os resultados obtidos a partir do banco de dados de ambas as aplicações. A Tabela 17 exibe as médias aritméticas e os desvios padrão calculados a partir de tais resultados para cada uma das grandezas monitoradas.

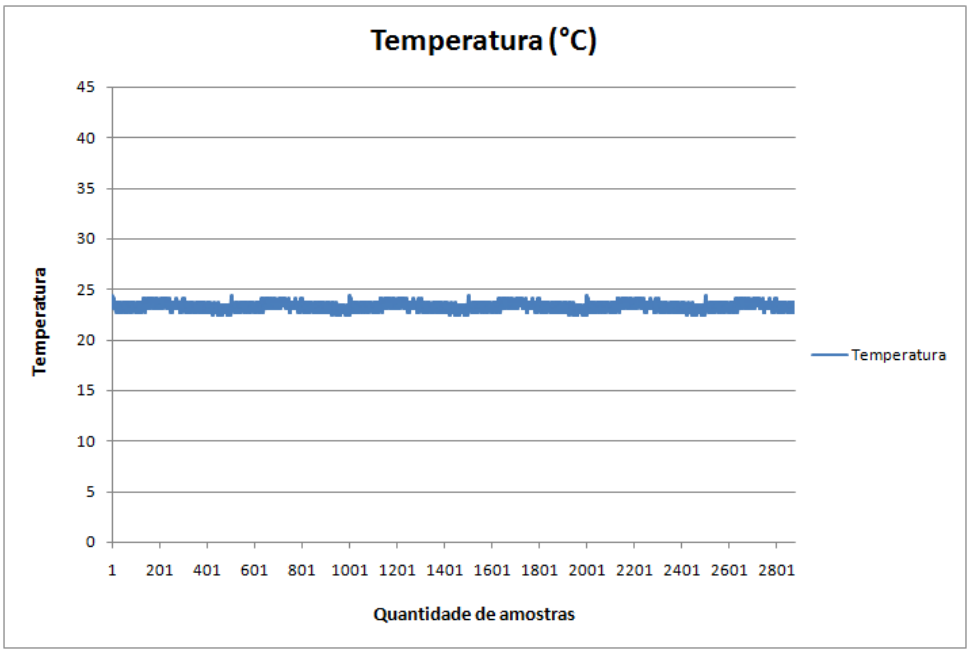
Tabela 17: Média e desvio padrão de cada grandeza monitorada.

Grandeza	Média	Desvio padrão
Temperatura	23,30	0,40
Luminosidade	100,06	34,63
RSSI Down	-56,84	1,41
RSSI Up	-56,82	1,40

Fonte: Elaboração própria.

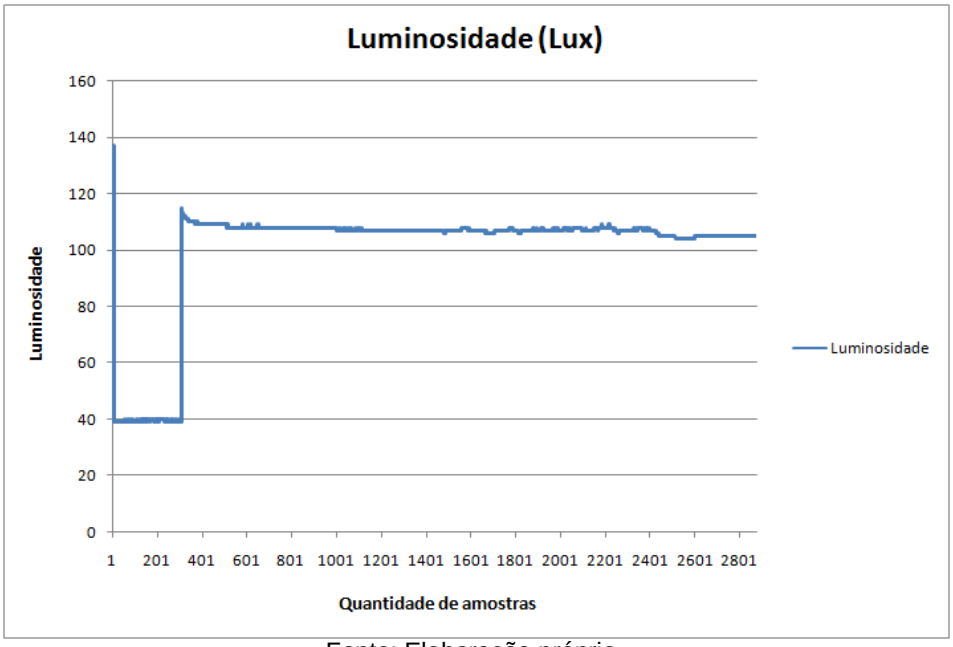
As Figuras 34 e 35 exibem respectivamente os gráficos referentes a temperatura e luminosidade, gerados a partir dos 2.880 valores retornados pelo comando SQL executado para cada uma dessas grandezas considerando o período do dia 06/04/2015.

Figura 34: Gráfico da temperatura medida no Publisher HTTP e no SMR.



Fonte: Elaboração própria.

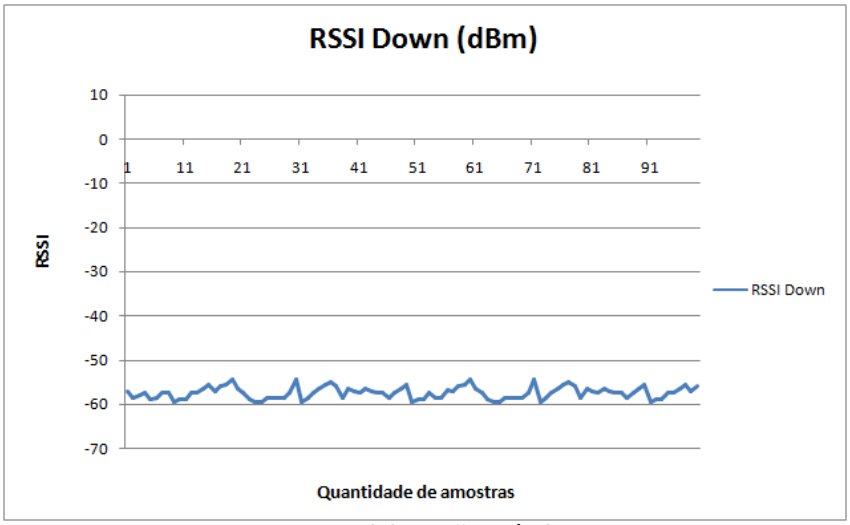
Figura 35: Gráfico da luminosidade medida no Publisher HTTP e no SMR.



Fonte: Elaboração própria.

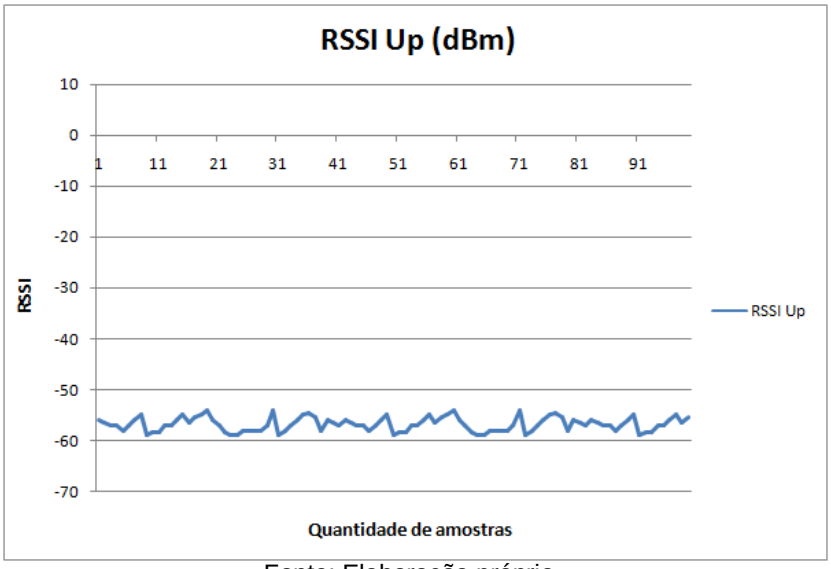
Levando-se em conta que a variação apresentada pelas medições obtidas das grandezas RSSI Up e RSSI Down foi muito pequena, optou-se pela geração dos gráficos correspondentes utilizando somente as 100 primeiras amostragens para que tais variações ficassem nítidas nos gráficos. As Figuras 36 e 37 exibem os gráficos relacionados a tais amostragens.

Figura 36: Gráfico de RSSI Down medido no Publisher HTTP e no SMR.



Fonte: Elaboração própria.

Figura 37: Gráfico de RSSI Up medido no Publisher HTTP e no SMR.



Fonte: Elaboração própria.

Do ponto de vista qualitativo o sistema também mostrou-se adequado, apresentando resultados coerentes de acordo com os dados que foram coletados em ambas as aplicações, ou seja, no *software* do *Publisher* HTTP e no SMR.

6.3.2. Cenário 2: Testando a confiabilidade do sistema

De acordo com a NBR ISO/IEC 9126, o item de qualidade confiabilidade, pode ser testado com relação à maturidade, tolerância a falhas e recuperabilidade do sistema em questão.

Neste cenário de teste, o sistema desenvolvido com base na arquitetura do *Publisher* HTTP foi testado levando-se em consideração a tolerância a falhas e a recuperabilidade.

Segundo a NBR ISO/IEC 9126, a tolerância a falhas demonstra a capacidade do *software* em manter o funcionamento adequado mesmo quando ocorrem falhas em uma ou mais partes do sistema. Já a recuperabilidade, representa a capacidade do sistema em se recompor após uma falha, restabelecendo seus níveis de desempenho e recuperando os seus dados.

Para este cenário de teste, foram utilizadas as mesmas definições de funcionamento do sistema e da RSSF, além das configurações apresentadas no cenário anterior.

6.3.2.1. Avaliando a Tolerância a Falhas

Este teste foi iniciado com o banco de dados de ambas as aplicações sem nenhum registro.

As operações de solicitação e envio foram iniciadas às 11:11:12 do dia 14/04/2015 e permaneceram em funcionamento até às 16:57:32 da mesma data, quando foram paradas, para que fosse verificada a quantidade de registros que cada um dos bancos de dados apresentava.

Neste momento os bancos de dados das duas aplicações apresentavam a mesma quantidade de registros, exatamente 2776.

Essa informação foi obtida através da linguagem SQL executando-se o comando exibido na Tabela 18 em ambos os bancos de dados:

Tabela 18: Quantidade de registros após execução normal.

Comando SQL	Quantidade de registros no Publisher HTTP	Quantidade de registros no SMR
select count(*) from MONITORAMENTO	2776	2776

Fonte: Elaboração própria.

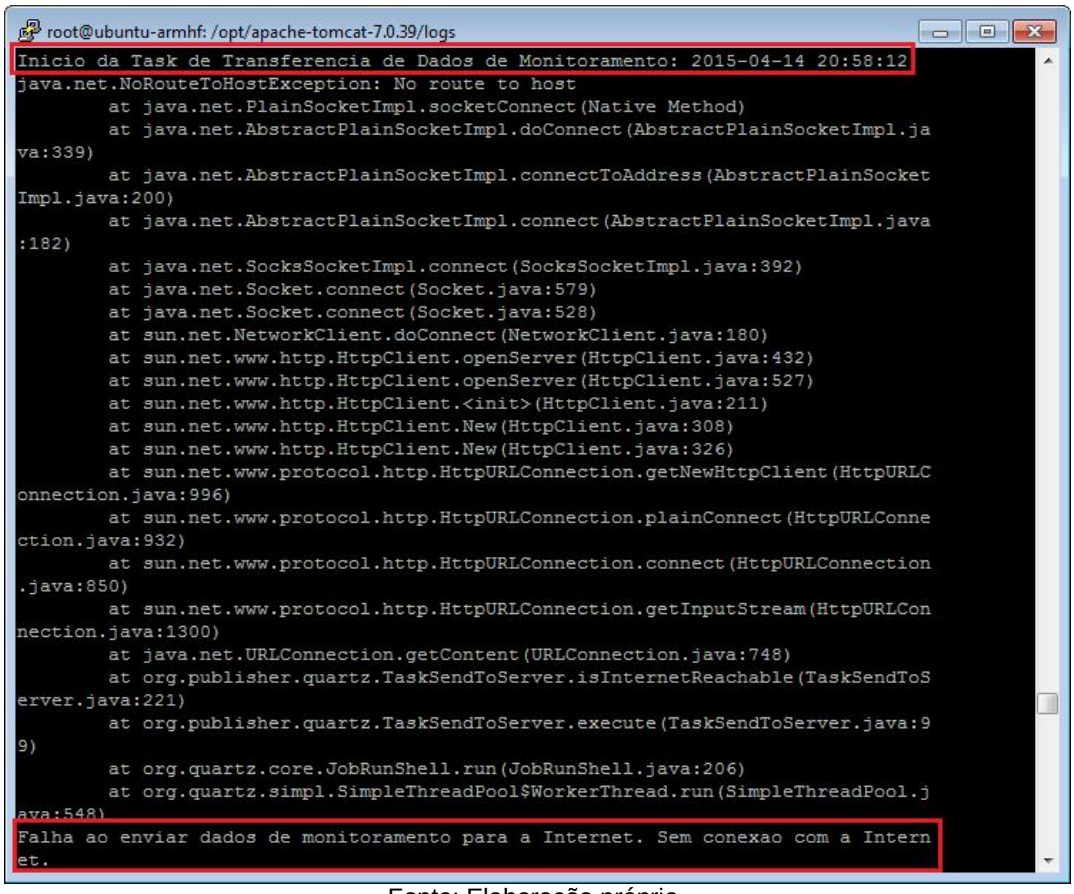
Para que uma falha fosse provocada no sistema, o cabo RJ-45 que ligava o Publisher HTTP à intranet, e que permitia a ele acessar a Internet, foi desconectado, como mostra a Figura 38.

Figura 38: Publisher HTTP desconectado fisicamente da intranet.

Fonte: Elaboração própria.

As operações de solicitação e envio foram reiniciadas às 17:47:03 da data em questão, 14/04/2015. Notou-se uma falha no envio das informações do *Publisher* HTTP para o SMR devido a falta de conexão com a Internet, conforme a Figura 39 exibe.

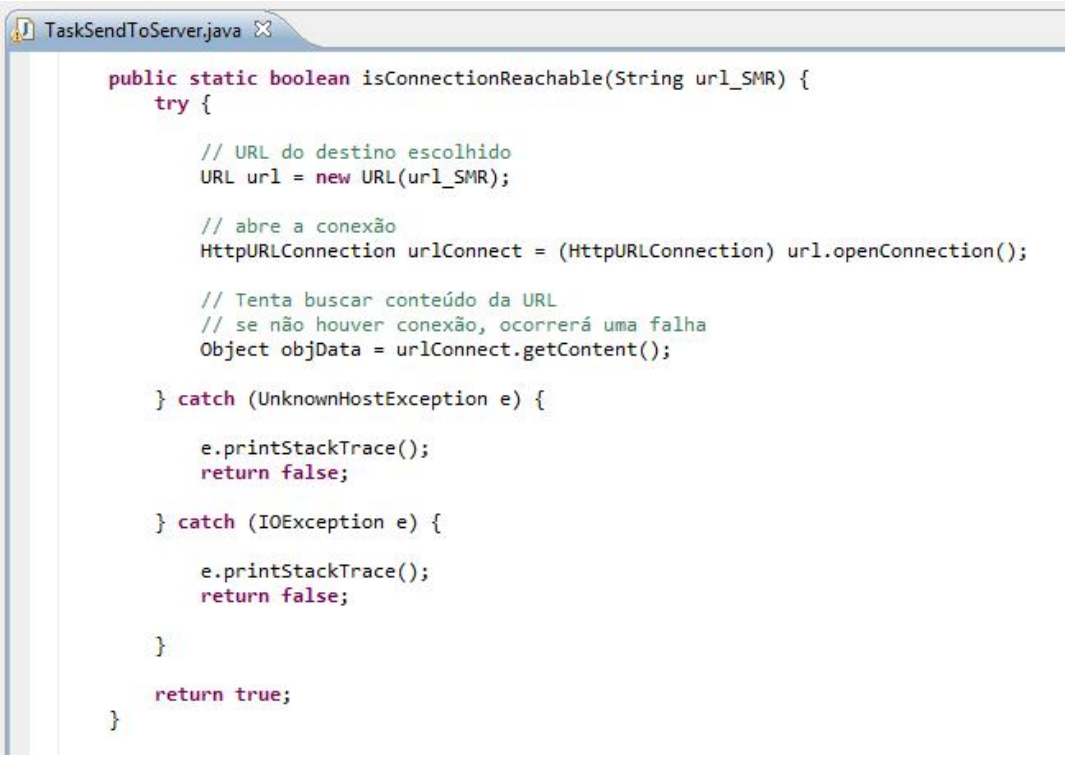
Figura 39: Falha apresentada pelo *Publisher* HTTP ao tentar acessar a Internet.



Fonte: Elaboração própria.

A mensagem de erro foi emitida com base no trecho de código-fonte do *Publisher* HTTP exibido na Figura 40. Nesse caso, a conexão com a *Internet* foi verificada buscando-se o *Uniform Resource Identifier* (URI) do *SMR*, *http://publisherhttpsmr-env.elasticbeanstalk.com/*, fornecido como parâmetro à função *isConnectionReachable*.

Figura 40: Código-fonte do *Publisher* HTTP que testa a conectividade com a Internet.



Fonte: Elaboração própria.

Após 3 horas 48 minutos e 29 segundos de funcionamento, o sistema foi parado novamente para que pudesse ser verificado que o *Publisher* HTTP manteve-se operante mesmo após o início da falha provocada, impedindo-o de enviar as informações provenientes da RSSF para o SMR.

Essa verificação comprovou que, mesmo sem acesso à Internet, o *Publisher* HTTP continuou solicitando as informações à RSSF e armazenando-as em seu banco de dados local.

Uma nova consulta foi realizada no banco de dados de ambas as aplicações, onde foram identificados 1824 registros a mais no banco de dados do Publisher

HTTP em relação ao banco de dados do SMR, conforme pode ser observado na Tabela 19.

Tabela 19: Quantidade de registros após execução sem envio de dados ao SMR.

Comando SQL	Quantidade de registros no Publisher HTTP	Quantidade de registros no SMR
select count(*) from MONITORAMENTO	4600	2776

Fonte: Elaboração própria.

Isso comprova que o sistema baseado no *Publisher* HTTP possui uma tolerância a falhas adequada, uma vez que, mesmo sem poder enviar as informações para o sistema que se encontra na Internet, os dados continuaram a ser coletados junto à RSSF, permitindo assim, que eles fossem enviados ao SMR quando a conexão com a Internet fosse restabelecida.

6.3.2.2. Avaliando a Recuperabilidade do Sistema

Este teste diz respeito a recuperabilidade do sistema, e dá continuidade ao teste iniciado no item anterior.

O intuito é demonstrar que, mesmo após a falha provocada anteriormente, o sistema é capaz de recuperar o seu funcionamento normal, seus níveis de desempenho e os seus dados.

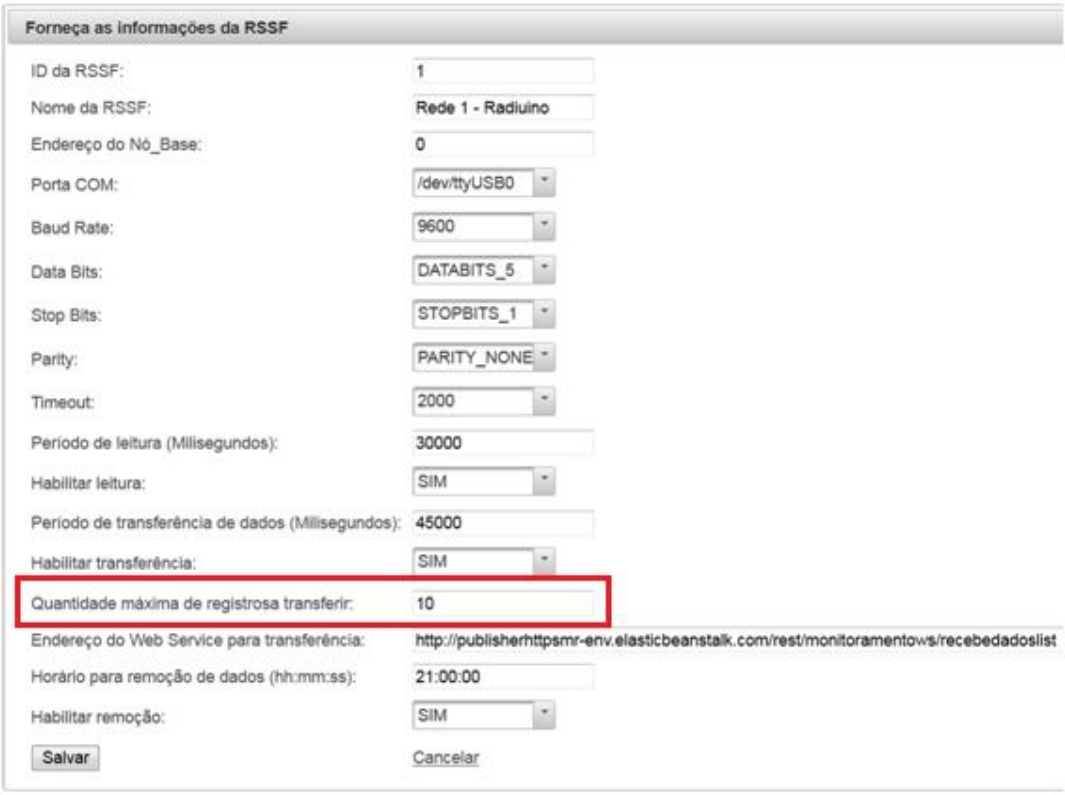
Após a constatação de que havia uma diferença de 1824 registros a mais no banco de dados do *Publisher* HTTP em relação ao banco de dados do SMR, o hardware do *Publisher* HTTP foi conectado novamente à intranet, permitindo que os dados fossem enviados via Internet para o SMR.

As operações do Publisher HTTP, com acesso à Internet, foram reiniciadas às 21:50:00 do dia 14/04/2015.

As mesmas configurações utilizadas nos testes anteriores foram mantidas e utilizadas para este teste. Ou seja, o *Publisher* HTTP solicitando dados à RSSF de 30 em 30 segundos, e enviando tais informações ao SMR a cada 60 segundos.

No entanto, é necessário destacar a utilização do parâmetro de configuração que define o número máximo de registros que podem ser enviado do *Publisher* HTTP para o SMR. Esse parâmetro pode ser visualizado em destaque na Figura 41.

Figura 41: Parâmetros de configuração do Publisher HTTP.



Fonte: Elaboração própria.

Esse parâmetro é utilizado para selecionar uma determinada quantidade de registros no banco de dados do *Publisher* HTTP, que serão enviados ao SMR. A existência de um parâmetro de configuração como este é importante, pois, de outra forma o *Publisher* HTTP tentaria enviar ao SMR todos os registros presentes em seu banco de dados, que possuíssem o status de não-enviado. Isso provavelmente, levaria a uma falha do sistema, principalmente se for considerado o teste atual, onde tem-se 1824 registros não-enviados e o *Publisher* HTTP rodando em um hardware com recursos limitados. Para todos os testes apresentados por este trabalho foi utilizado o número máximo de 20 registros.

Numa situação de operação normal do sistema, a cada 30 segundos é feita uma solicitação de leitura à RSSF gerando 4 registros no banco de dados do

Publisher HTTP. Como o envio ao SMR ocorre a cada 60 segundos, os últimos 8 registros gerados no banco de dados do *Publisher* HTTP são enviados.

Na situação que está sendo considerada para demonstração dos resultados relacionados à recuperabilidade do sistema, o *Publisher* HTTP continuou efetuando solicitações à RSSF e enviando tais dados ao SMR. Além dos dados provenientes de leituras recentes, deveriam ser enviados também ao SMR os 1824 registros presentes somente no banco de dados do *Publisher* HTTP, para que ambos passassem a armazenar as mesmas informações.

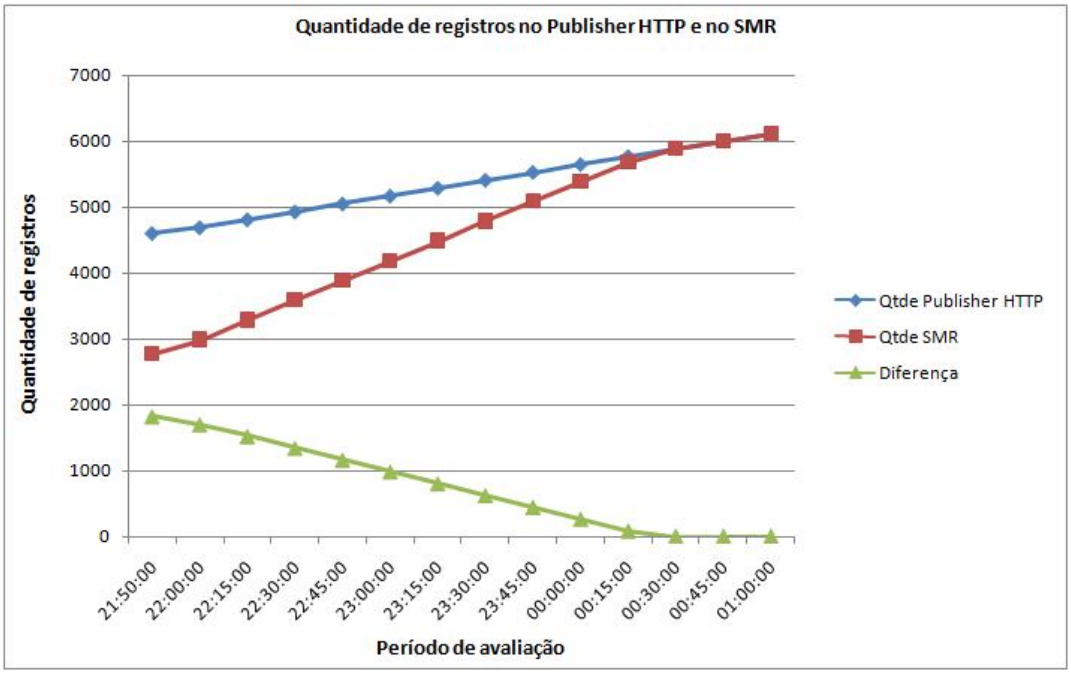
Respeitando-se o limite máximo de 20 registros por envio de dados ao SMR, observou-se que o *Publisher* HTTP, a partir do reinício de suas operações às 21:50:00 do dia 14/04/2015, levou aproximadamente 2 horas e 32 minutos para transferir os 1824 registros para o SMR.

Por volta de 00:22:00 do dia 15/04/2015 o conteúdo de ambas as aplicações passaram a armazenar a mesma quantidade de registros, cerca de 5816.

Além dos 1824 registros enviados pelo mecanismo de recuperabilidade, o Publisher HTTP enviou cerca de 1216 registros referentes a novas solicitações realizadas à RSSF. A cada envio realizado pelo Publisher HTTP ao SMR, 4 registros eram referentes a novas leituras e 6 provenientes dos 1824 que se encontravam somente no banco de dados do *Publisher* HTTP.

A Figura 42 exibe o período de transferência de dados analisado com as quantidades dos bancos de dados de ambas as aplicações e a diferença entre eles. Foram incluídos mais dois horários de verificação, 00:45:00 e 01:00:00, para demonstrar que as quantidades de ambos os bancos de dados permaneceram iguais a partir de 00:30:00.

Figura 42: Gráfico de recuperação de registros.



Fonte: Elaboração própria.

Dessa forma, conclui-se que em termos de recuperabilidade, o sistema possui um funcionamento adequado. Após a falha que desestabilizou o sistema causando uma diferença de conteúdo entre os bancos de dados do *Publisher* HTTP e do SMR, o sistema recuperou-se assim que a conexão com a Internet foi restabelecida. Não somente novas informações foram enviadas ao sistema localizado na Internet, mas também os dados de solicitações antigas, localizados somente no banco de dados do *Publisher* HTTP, equalizando o conteúdo dos bancos de dados das duas aplicações.

6.3.3. Cenário 3: Testando a Eficiência do Sistema

A eficiência é um item de qualidade da NBR ISO/IEC 9126 que pode ser testado considerando o comportamento do sistema em relação ao tempo, e também em relação à utilização dos recursos tanto do *hardware* quanto do sistema operacional.

Considerando que o objetivo principal apresentado por este trabalho é a integração de RSSFs com a Internet através da tecnologia de *web services*, avaliou-

se a eficiência somente do recurso que consiste na base dessa integração, no caso, o *web service RecebeDadosMonitoramentoWs* implementado no SMR.

Dessa forma, esse teste foi aplicado somente na parte do sistema que corresponde ao SMR, disponibilizado por um servidor virtual na Internet, suportado por uma estrutura de computação em nuvem. O intuito foi verificar o tempo de resposta desse *web service* em relação às solicitações enviadas a partir do software cliente, no caso, o *software* que compõe a UI do *Publisher* HTTP.

Com base num modelo de qualidade para web services (OASIS - WSQM, 2011) criado pelo consórcio OASIS, *Advancing open standards for the information society* (OASIS, 2015), avaliou-se o tempo de resposta ou response time do web service. Segundo o modelo de qualidade OASIS (OASIS - WSQM, 2011), o tempo de resposta ou response time refere-se ao tempo de duração entre o envio da requisição do software cliente ao *web service* até o momento do recebimento da resposta. O tempo de resposta do *web service* pode ser calculado subtraindo-se o horário da requisição do horário em que o software cliente recebe a resposta do web service, confome podemos observar na fórmula exibida pela Figura 43.

Figura 43: Fórmula para calcular tempo de resposta.

Tempo de resposta = Horário de recebimento da resposta – Horário da requisição

Fonte: Elaboração própria.

O tempo de resposta do *web service RecebeDadosMonitoramentoWs* foi avaliado a partir da execução do próprio software que representa a UI do *Publisher* HTTP conforme as configurações apresentadas nos testes anteriores. Ou seja, as solicitações à RSSF feitas de 30 em 30 segundos, e o envio ao SMR realizado a cada 60 segundos de um conteúdo referente a 8 registros de dados do *Publisher* HTTP. Foram adicionadas ao código-fonte do *software* da UI linhas de comando que registram a data e a hora no momento do envio da requisição ao *web service* e no seu retorno, conforme podemos observar na Figura 44. Essas informações foram gravadas no arquivo de registro de atividades do Apache Tomcat e posteriormente analisadas para a obtenção dos resultados desse teste.

Figura 44: Trecho de código-fonte que registra envio e retorno do *web service*.

```
System.out.println("RecebeDadosMonitoramentoWs - requisicao enviada: "
        + sdf.format( new Date()));

//Chamada ao web service RecebeDadosMonitoramentoWs e recebimento da resposta
HttpResponse response = client.execute(post);

System.out.println("RecebeDadosMonitoramentoWs - resposta recebida: "
        + sdf.format( new Date()));
```

Fonte: Elaboração própria.

Foram analisados os tempos de requisição e resposta referentes a 720 envios de dados do Publisher HTTP ao *web service RecebeDadosMonitoramentoWs*, correspondendo a 12 horas de execução do sistema, na data de 06/04/2015.

Os tempos de resposta foram calculados para cada uma das chamadas com base na fórmula apresentada anteriormente e em seguida, foi obtida a média do tempo de resposta do *web service RecebeDadosMonitoramentoWs* que foi de 1,25 segundos com um desvio padrão de 0,12 segundos.

Para efeito de comparação, foi reproduzido numa rede local de 100 Mbps, o mesmo ambiente onde o SMR estava instalado na Internet. A Tabela 20 exibe os softwares utilizados para compor tal ambiente e suas respectivas versões.

Tabela 20: Softwares utilizados no ambiente de teste de performance.

Tipo de software	Software	Versão
Sistema Operacional	Ubuntu Linux Server	14.04
Container Web	Apache Tomcat	7
Banco de dados	Apache Derby	10.11
Runtime Java	Java Virtual Machine	7

Fonte: Elaboração própria.

Como hardware para o servidor foi utilizado um microcomputador dotado de um processador Intel Core i3 de 2.20 GHz de processamento, 6,00 GB de memória RAM e um disco rígido de 500 GB.

O tempo de resposta médio do *web service RecebeDadosMonitoramentoWs* foi calculado a partir dos tempos de requisição e resposta dos envios de dados executados pelo *Publisher* HTTP ao SMR, porém, na rede local. No momento dos testes, somente o *Publisher* HTTP e o servidor onde o SMR foi instalado faziam uso da rede local. Considerou-se a mesma quantidade de amostras do teste inicial, 720. Este teste foi realizado em 08/04/2015.

Nesse caso, a média do tempo de resposta do *web service RecebeDadosMonitoramentoWs* foi de 1,21 segundos com um desvio padrão de 0,10 segundos.

A Tabela 21 compara os valores de tempo de resposta do web service *RecebeDadosMonitoramentoWs*, obtidos nos testes realizados com o SMR *Internet* e na rede local.

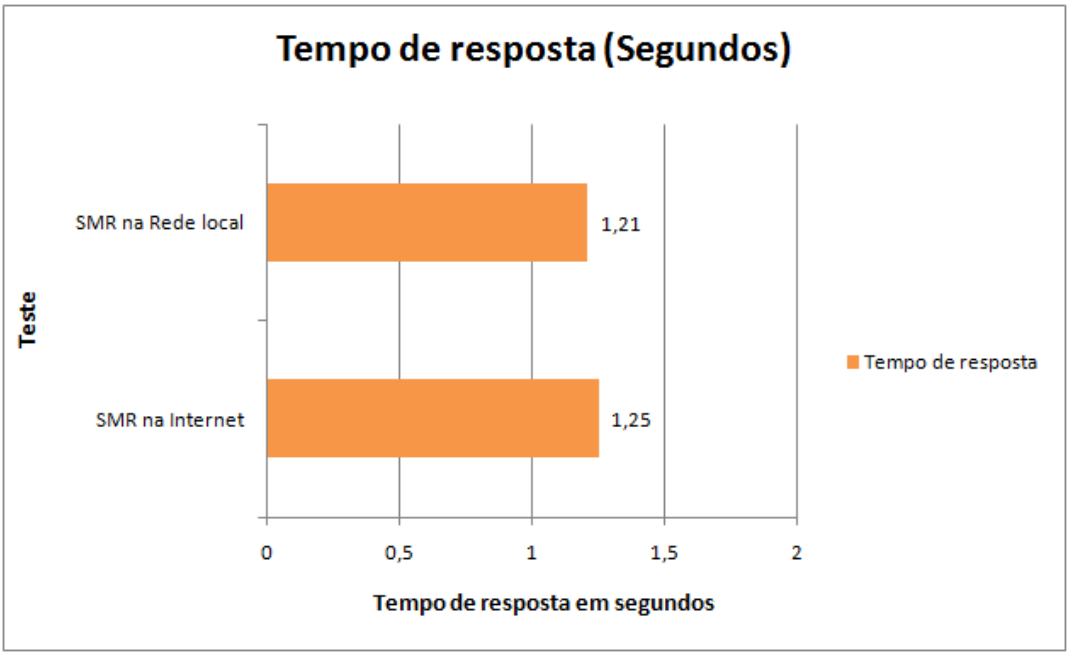
Tabela 21: Média de tempo de resposta do web service.

Média do tempo de resposta (Segundos)	
SMR na Internet	SMR na Rede local
1,25	1,21

Fonte: Elaboração própria.

Conforme podemos observar na Figura 45, o desempenho do web service na Internet pode ser considerado satisfatório em relação ao teste realizado na rede local, uma vez que a diferença apresentada foi de apenas 0,04 segundos.

Figura 45: Média de tempo de resposta do *web service* numa *intranet* e na *Internet*.



Fonte: Elaboração própria.

Para medir o quanto a quantidade de registros enviada ao SMR impactava no tempo de resposta do *web service RecebeDadosMonitoramentoWs*, foram realizados testes de envio com diferentes quantidades de registros.

Nesse caso específico, o envio dos dados ocorreu somente para o SMR localizado no *host* da Internet.

O teste realizado anteriormente, foi baseado no envio de 4 registros de dados da tabela Monitoramento, do banco de dados do *Publisher* HTTP.

Os demais testes realizados consideraram para o envio as quantidades 8, 20, 50 e 100 registros da tabela Monitoramento para o SMR.

A Tabela 22 exibe as quantidades de registros enviadas ao SMR e as médias de tempo de resposta obtidas em cada situação. Todas as médias foram calculadas com base em 50 amostragens obtidas para cada situação.

Tabela 22: Média de tempo de resposta em relação à quantidade de registros enviada.

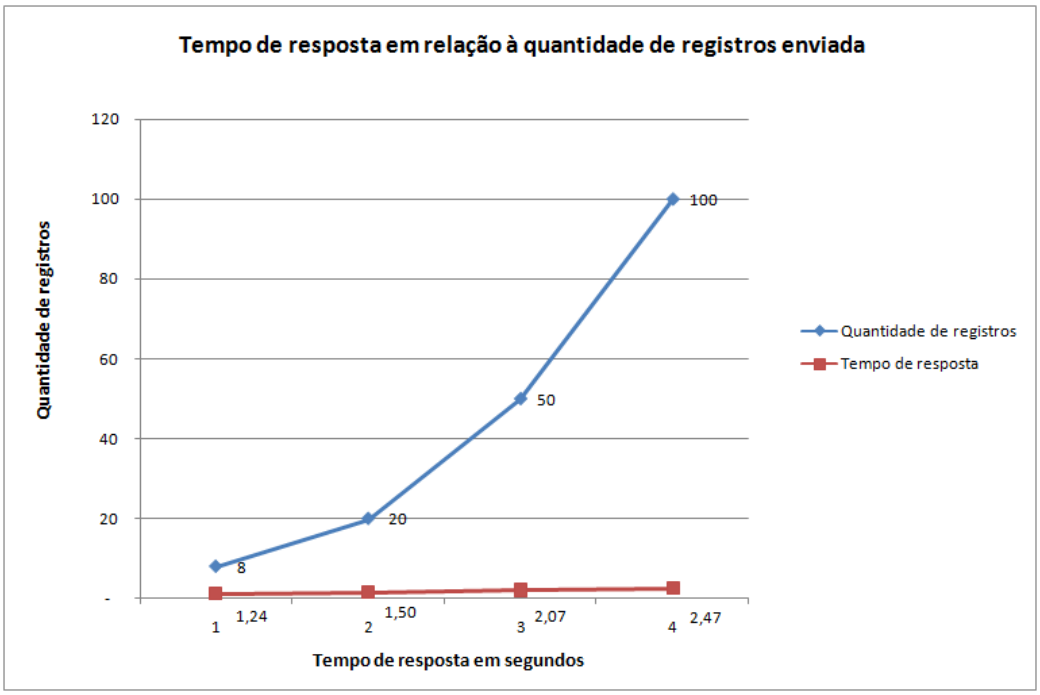
Quantidade de Registros Enviada	Tempo de Resposta (Segundos)
8	1,24
20	1,50
50	2,07
100	2,47

Fonte: Elaboração própria.

Observou-se que a quantidade de registros enviada ao SMR impacta diretamente no tempo de resposta do *web service*. Uma vez que o *web service*, ao receber tais informações, deve persisti-las num banco de dados, conforme aumenta a quantidade de registros enviados, aumenta proporcionalmente o número de operações de persistências realizadas no banco de dados, e também o tempo de resposta.

Essa proporção entre o aumento do número de registros enviados e o tempo de resposta, pode ser observado no gráfico exibido na Figura 46.

Figura 46: Aumento do tempo de resposta em função do aumento de registros enviados



Fonte: Elaboração própria.

6.4. Resultados

De acordo com os resultados dos testes apresentados anteriormente, observou-se que é possível a interconexão de RSSFs com a Internet utilizando-se a tecnologia de *web services* sobre o protocolo HTTP.

Foi demonstrado que a arquitetura proposta por este trabalho permite a interconexão entre esses dois padrões de rede sem que haja alterações na pilha de protocolos de nenhuma delas.

Os testes apresentaram ainda, resultados satisfatórios, dentro dos itens de qualidade considerados, validando a implementação de baixo custo proposta e a utilização de recursos de computação em nuvem, permitindo maior flexibilidade e escalabilidade para o sistema.

7. CONCLUSÃO

Este trabalho teve como proposta uma arquitetura para interconexão de RSSFs com a Internet utilizando-se a tecnologia de *web services* sobre o protocolo HTTP, de forma a estender ao máximo a acessibilidade aos dados captados pelas RSSFs. O principal intuito foi prover uma solução de interconexão que atuasse somente na camada de aplicação sem que a pilha de protocolos de ambas as redes precisasse ser alterada. De forma complementar buscou-se uma implementação de baixo custo tanto em termos de hardware quanto de software.

A implementação da arquitetura proposta com a utilização de recursos *open-hardware* e *open-source* se mostrou viável, contribuindo para o baixo custo da solução.

Por fim, a utilização do SMR num ambiente de computação em nuvem permite que o monitoramento da RSSF e dos dados captados por ela seja efetuado a qualquer momento, independente da localização física do administrador do sistema. O suporte de um ambiente de computação em nuvem garante ainda maior flexibilidade e escalabilidade ao sistema, permitindo seu crescimento sem que haja a preocupação com a limitação de recursos.

Em relação à trabalhos futuros, é essencial que o sistema implementado seja testado não somente com um maior número de sensores, mas também com várias RSSFs de diferentes padrões sendo gerenciadas a partir do SMR, conforme proposto por este trabalho. Isto tornaria a proposta mais próxima de sistemas reais.

Estender as funcionalidades do SMR para controle das RSSFs também é um item importante a ser desenvolvido, uma vez que a implementação demonstrada contempla somente o monitoramento da RSSF e dos dados por ela captados.

REFERÊNCIAS

ALVES, E. C. S. Avaliação de Parâmetros de Desempenho de Rede de Sensores Sem Fio em Casa de Vegetação para Cultivo Hidropônico de Morangos. Dissertação (Mestrado em Engenharia Agrícola). Faculdade De Engenharia Agrícola, Universidade Estadual de Campinas, Campinas, SP, 2011.

AWS. Site da empresa Amazon Web Services. Disponível em: <http://www.aws.com/>. Acesso em: 12 mar. 2015.

APACHE DERBY. Site do Projeto Apache Derby. Disponível em: db.apache.org/derby. Acesso em: 12 mar. 2015.

APACHE TOMCAT. Site do Projeto Apache Tomcat. Disponível em: <http://tomcat.apache.org/>. Acesso em: 12 mar. 2015.

APACHE. Site da Apache Software Foundation. Disponível em: <http://www.apache.org/>. Acesso em: 12 mar. 2015.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126: Engenharia de software - Qualidade de produto Parte 1: Modelo de qualidade. Rio de Janeiro, 2003.

BENAVENTE, J. C. C. Monitoramento ambiental de vinhedos utilizando uma rede de sensores sem fio que coleta dados com um intervalo de amostragem variável. Dissertação (Mestrado em Engenharia). Departamento de Engenharia da Computação e Sistemas Digitais, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, 2010.

BRANQUINHO, O. C.; “Plataforma Radiuino Para Estudo de Rede de Sensores Sem Fio”. Disponível em: <http://www.foruns.unicamp.br/Arquivos%20Bibioteca%20Virtual/Palestras/2110/Plataforma%20Radiuino%20v14.pdf>. Acesso em: 10 mar. 2015.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S. Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities. In: IEEE

International Conference on High Performance Computing and Communications, 10., Proceedings…, setembro, 2008.

CARVALHO, F. B. S. et al. Aplicações Ambientais de Redes de Sensores Sem Fio, Revista de Tecnologia da Informação e Comunicação, v. 2, p. 14-19, 2012.

CHEN, Y. et al. Integrated Wireless Access Point Architecture for Wireless Sensor Networks. In: Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT 2009), vol.1, 2009, Gangwon-Do, South Korea. Proceedings... [S.I.]: IEEE, 2009, p.713-718.

DAI, H.; HAN, R. Unifying Micro Sensor Networks with the Internet via Overlay Networking. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004. Proceedings... [S.I.]: IEEE, 2004, p. 571-572.

DAMASO, A. V. L.; DOMINGUES, J. P. O.; ROSA, N. S. SAGe: Sensor Advanced Gateway for Integrating Wireless Sensor Networks and Internet, 2010.

DUNKELS, A. et al. Connecting wireless sensornets with TCP/IP networks. In: Proceedings of the 2nd International Conference on Wired/Wireless Internet Communications (WWIC '04), 2004, Frankfurt, Germany. Proceedings... [S.I.]: Springer Berlin Heidelberg, 2004, p. 143-152.

ECLIPSE. Site do Projeto Eclipse. Disponível em: <http://www.eclipse.org/>. Acesso em: 12 mar. 2015.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. 76 p. Ph.D. dissertation (Doctor of Philosophy in Information and Computer Science). University of California, Irvine, CA, USA, 2000.

GOOGLE. Site do Google Apps for business. Disponível em: <http://www.google.com/apps/intl/en/business/index.html>. Acesso em: 15 mar 2015.

GUBBIA, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (2013) 1645–1660, Elsevier.

H2 DATABASE PERFORMANCE. Estudo sobre a performance dos bancos de dados Java que podem rodar de forma auto-contida em aplicações do tipo web. Disponível em: <http://www.h2database.com/html/performance.html>. Acesso em: 10 ago. 2014.

H2. Site do Projeto H2. Disponível em: www.h2database.com. Acesso em: 12 mar. 2015.

HAMAD, H.; SAAD, M.; ABED, R. "Performance Evaluation of RESTful Web Services for Mobile Devices". International Arab Journal of e-Technology, Vol. 1, No. 3, Janeiro, 2010.

HO, M.; FALL, K. Delay tolerant networking for sensor networks, Proc. IEEE Conf. Sensor and Ad Hoc Comm. and Networks, 2004.

HSQLDB. Site do Projeto HSQLDB. Disponível em: www.hsqldb.org. Acesso em: 12 mar. 2015.

HUI, J. W.; CULLER, D. E. Extending IP to Low-Power, Wireless Personal Area Networks. IEEE Internet Computing, IEEE, USA, Volume: 12 , Issue: 4, p. 37-45, 2008.

IBM. Site do IBM BlueMix. Disponível em: <http://www.ibm.com/cloud-computing/bluemix/>. Acesso em: 15 mar 2015.

IEEE COMPUTER SOCIETY. Guide to the Software Engineering Body of Knowledge. Piscataway, NJ, 2014.

IEEE, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), 2011.

JETTY. Site do Projeto Jetty. Disponível em: <http://www.eclipse.org/jetty/>. Acesso em: 12 mar. 2015.

KARL, H.; WILLIG, A. Protocols and Architectures for Wireless Sensor Networks. West Sussex, England: John Wiley& Sons Ltd, 2005. 78-79 p.

KIM, J.-H. et al. Address internetworking between WSNs and internet supporting web services. In: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE '07), 2007, Seoul, South Korea. Proceedings... [S.I.]: IEEE Computer Society, 2007, p. 232–237.

KIM, J.-H. et al. Address internetworking between WSNs and internet supporting web services. In: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE '07), 2007, Seoul, South Korea. Proceedings... [S.I.]: IEEE Computer Society, 2007, p. 232–237.

LEI, S. et al. Connecting Sensor Networks with TCP/IP Network, in Proc. of the International Workshop on Sensor Networks (IWSN 2006), Harbin, China, Jan. 2006, pp. 330-334.

LI, L.; YANG, S.; WANG, L.; GAO, X. "The Greenhouse Environment Monitoring System Based on Wireless Sensor Network Technology", Proceedings of the 2011 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, March 20-23, 2011, Kunming, China.

LUDOVICI, A.; CALVERAS, A.; J. CASADEMONT, J. Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network, em Wireless Network Group (WNG), Mòdul, 2011.

MADEIRA, J. F. L. GERÊNCIA DE REDES SENSORES SEM FIO COM SNMP: UMA ABORDAGEM COM PROXY GATEWAY. Dissertação (Mestrado em Engenharia Elétrica). CEATEC - Faculdade de Engenharia Elétrica, Pontifícia Universidade Católica de Campinas, Campinas, SP, 2014.

MONTENEGRO, N. et al. Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944, 2007.

MORAIS, R., FERNANDES, M. A., MATOS, S. G., SERÔDIO, C., FERREIRA, P.J.S.G., REIS, M.J.C.S. A Zigbee Multi-Powered Wireless Acquisition Device for Remote Sensing Applications in Precision Viticulture. Computers and Electronics in Agriculture. v. 62, n. 2, p. 94-106, 2008.

NAYAK, A.; STOJMENOVIC, I. Wireless Sensor and Actuator Networks. John Wiley & Sons, New Jersey, USA, 2010.

OASIS - WSQM. Web. OASIS Web Services Quality Model Technical Comitee. 2011. Disponível em: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm>. Acesso em: 12 mar. 2015.

OASIS. Web site oficial do consórcio OASIS - Advancing open standards for the information society. Disponível em: <http://www.oasis-open.org/ >. Acesso em: 12 mar. 2015.

ORACLE J2EE. Site do Tutorial do Java Enterprise Edition 1.4. 2005. Disponível em: <http://docs.oracle.com/javaee/1.4/tutorial/doc/WebApp.html>. Acesso em: 17 fev. 2014.

ORACLE JAVA. Site Oficial da Linguagem de Programação Java. Disponível em: http://www.java.com. Acesso em: 20 abr. 2014.

PHP. Site do Projeto PHP. Disponível em: < http://php.net/>. Acesso em: 12 mar. 2015.

PLUMBR. Artigo sobre os web containers mais populares da atualidade. 2013. Disponível em: < https://plumbr.eu/blog/most-popular-java-environments >. Acesso em: 10 ago. 2014.

PLUMBR. Artigo sobre os web containers que consomem menos memória . 2012. Disponível em: < https://plumbr.eu/blog/most-popular-java-environments >. Acesso em: 10 ago. 2014.

POTTI, P. K.; AHUJA, A.; UMAPATHY K. Comparing Performance of Web Service Interaction Styles: SOAP vs. REST. In: Proceedings of the Conference on Information Systems Applied Research, New Orleans Louisiana, USA, 2012.

RADIO IT. Site da empresa Radio IT. Disponível em: < http://radioit.com.br/>. Acesso em: 17 jan. 2015.

RADIUINO. Site do Projeto Radiuino. Disponível em: <http://www.radiuino.cc>. Acesso em: 22 abr. 2014.

RAJESH, V. et Al. Integration of Wireless Sensor Network with Cloud, International Conference on Recent Trends in Information, Telecommunication and Computing, p. 321–323, March 2010.

RASPBERRY PI. Site do Projeto Raspberry Pi. Disponível em: <http://www.raspberrypi.org>. Acesso em: 10 fev. 2014.

RASPBERRY PI. Site do Projeto Raspberry Pi. Disponível em: <http://www.raspberrypi.org>. Acesso em: 10 fev. 2014.

RICHARDSON, L.; RUBY, S. RESTful Web Services, United States of America, O'Reilly, 2007.

SCADABR. Site do Projeto ScadaBR. Disponível em: www.scadabr.org.br. Acesso em: 12 mar. 2014.

SHU, L.; WU, X.; HU, H. Connecting heterogeneous sensor networks with IP based wire, wireless networks. In: Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006, Gyeongju, South Korea. Proceedings... [S.I.]: IEEE, 2006, p. 27-28.

SOHRABY, K.; MINOLI, D.; ZNATI,T. WIRELESS SENSOR NETWORKS. Technology, Protocols, and Applications, United Kingdom, John Wiley & Sons Ltd, 2007.

SOUSA, M. P.; LOPES, W. T. A. Desafios em Redes de Sensores sem Fio. Revista De Tecnologia da Informação e Comunicação, Campina Grande; Vol.1, p. 41-47, 2011.

SUZUMURA, T. et al. Performance Comparison of Web Service Engines in PHP, Java, and C em IEEE International Conference on Web Services, 2008, Beijing, China. Proceedings... [S.I.]: IEEE Computer Society, 2008, p. 385–392.

TAURION, C. Cloud computing – computação em nuvem: transformando o mundo da tecnologia da informação. Rio de Janeiro: Brasport, 2009.

TING, H.; XIAOYAN, C.; YAN, Y. A new interconnection scheme for WSN and IPv6-based internet. In: Computing and Telecommunication, 2009. YC-ICT '09. IEEE

Youth Conference on Information, 2009, Beijing, China. Proceedings... [S.I.]: IEEE, 2009, p. 34–37.

USHALNAGAR, K.; MONTENEGRO, N.; SCHUMACHER,G. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement and Goals, RFC 4919, 2007.

VOORSLUYS, W.; BROBERG, J.; BUYYA, R. Introducing to cloud computing. In: BUYYA, Rajkumar; BROBERG, James; GOSCINSKY, Andrzej. Cloud computing principles and paradigms, New Jersey: Wiley Press, 2011.

W3C HTTP, Especificação do protocolo Hypertext Transfer Protocol - HTTP/1.1. 1999. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Acesso em: 05 nov. 2013.

W3C HTTP, Especificação do protocolo Hypertext Transfer Protocol - HTTP/1.1. 1999. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Acesso em: 05 nov. 2014.

W3C SOAP. Página da especificação do Simple Object Access Protocol (SOAP). 2007. Disponível em: <http://www.w3.org/TR/soap>. Acesso em: 10 fev. 2015.

W3C SUB. Página de submissão do Simple Object Access Protocol (SOAP). 2000. Disponível em: <http://www.w3.org/Submission/2000/05/>. Acesso em: 15 fev. 2015.

W3C WS. Página da especificação da Web Service Architecture. 2004. Disponível em: <http://www.w3.org/TR/ws-arch>. Acesso em: 10 fev. 2015.

WANG, W. D. et al. A Web Service based Gateway Architecture for Wireless Sensor Networks. In: 11th International Conference on Advanced Communication Technology (ICACT'2009), 2009, Gangwon-Do, South Korea. Proceedings... [S.I.]: IEEE, 2009, Volume 02, p. 1160 - 1163.

APÊNDICE A: DESCRIÇÃO DAS TABELAS QUE COMPÕEM O BANCO DE DADOS DO PUBLISHER HTTP.

Tabela 23: Detalhamento da tabela de dados Sensor.

Tabela	Sensor	
Descrição	Armazena os dados dos sensores que fazem parte da RSSF.	
Campo	Descrição	Tipo de dado
IdRede	Código de identificação da RSSF.	Integer
IdSensor	Código de identificação do Sensor.	Integer
NomeSensor	Nome para identificação do Sensor.	Varchar(50)

Fonte: Elaboração própria.

Tabela 24: Detalhamento da tabela de dados Grandeza.

Tabela	Grandeza	
Descrição	Armazena os dados das grandezas monitoradas pelo sistema.	
Campo	Descrição	Tipo de dado
IdGrandeza	Código de identificação da Grandeza.	Integer
DescGrandeza	Nome para identificação da Grandeza.	Varchar(50)

Fonte: Elaboração própria.

Tabela 25: Detalhamento da tabela de dados GrandezaDataPoint.

Tabela	GrandezaDataPoint	
Descrição	Armazena os dados dos bytes do pacote Radiuino que correspondem a cada uma das grandezas monitoradas.	
Campo	Descrição	Tipo de dado
IdGrandeza	Código de identificação da Grandeza.	Integer
IndiceByte	Posição do byte no pacote Radiuino.	Integer
NameByte	Nome para identificação do byte no pacote Radiuino.	Varchar(50)

Fonte: Elaboração própria.

Tabela 26: Detalhamento da tabela de dados Monitoramento.

Tabela	Monitoramento	
Descrição	Armazena os dados das grandezas calculadas.	
Campo	Descrição	Tipo de dado
IdRede	Código de identificação da RSSF.	Integer
IdNoSensor	Código de identificação do Sensor.	Integer
IdGrandeza	Código de identificação da Grandeza.	Integer
DataHoraSoilicitacaoRSSF	Data e Hora de envio do pacote de solicitação à RSSF.	Timastamp
DataHoraRetornoRSSF	Data e Hora de envio do pacote de solicitação à RSSF.	Timastamp
DataHoraTransferenciaWeb	Data e Hora de envio do pacote de solicitação à RSSF.	Timastamp
FlagTransferido	Indica se o registro já foi enviado para o SMR. Valor S ou N.	Varchar(1)
ValorGrandeza	Valor da grandeza calculada.	Double

Fonte: Elaboração própria.

Tabela 27: Detalhamento da tabela de dados Rede.

Tabela	Rede	
Descrição	Armazena os dados de configuração da RSSF.	
Campo	Descrição	Tipo de dado
IdRede	Código de identificação da RSSF.	Integer
NomeRede	Nome para identificação da RSSF.	Varchar(50)
PortaCom	Porta de comunicação serial que será utilizada para conectar o Publisher HTTP ao nó-base.	Varchar(30)
IdNoBase	Endereço lógico utilizado pelo nó-base.	Integer
BaudRate	Velocidade de transmissão de dados que deve ser utilizada para envio de dados ao nó-base.	Integer
DataBits	Quantidade de bits que deve ser considerada em um pacote que será transmitido ao nó-base.	Integer
Parity	Informa se deve ou não haver verificação de erro.	Integer
StopBits	Informação utilizada para sinalizar o fim da transmissão de um único pacote ao nó-base.	Integer
Timeout	Tempo limite para aceitação da conexão com o nó-base.	Integer
HabilitarLeitura	Campo Sim/Não utilizado para habilitar ou desabilitar a função de solicitação de leitura enviada à RSSF.	Varchar(3)
PeriodoLeitura	Freqüência com que a solicitação de leitura deve ocorrer. Valor em milisegundos.	Integer
HabilitarTransferencia	Campo Sim/Não utilizado para habilitar ou desabilitar a função de envio de dados ao SMR.	Varchar(3)
PeriodoTransferencia	Freqüência com que o envio de dados deve ocorrer. Valor em milisegundos.	Integer
HabilitarRemocao	Campo Sim/Não utilizado para habilitar ou desabilitar a função de remoção de registros já enviados para o SMR.	Varchar(3)
HorarioRemocao	Horário em que deve ocorrer a remoção dos registros já enviados ao SMR. Formato hh:mm:ss.	Varchar(8)

Fonte: Elaboração própria.