UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA

HUGO DE MELLO DANTAS

SiMoni: um Sistema de Monitoramento de ambientes de trabalho

Prof[a]. Silvana Rossetto, D.Sc.
Orientador

Rio de Janeiro, Dezembro de 2017

SiMoni: um Sistema de Monitoramento de ambientes de trabalho

Hugo de Mello Dantas

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Hugo de Mello Dantas

Aprovado por:

_____

Prof[a]. Silvana Rossetto, D.Sc.

_____

Prof. Gabriel Pereira da Silva, D.Sc.

_____

Prof. Claudio Miceli de Farias, D. Sc

RIO DE JANEIRO, RJ - BRASIL

Dezembro de 2017

Agradecimentos

Sendo Deus a causa primária de todas as coisas, direciono em primeiro lugar os agradecimentos que faço a Ele. Gratidão esta não apenas pelo sucesso deste trabalho, mas por ter Ele me dado nesta vida tantos motivos de alegria e tantas oportunidades de progresso.

Agradeço aos meus pais, à minha irmã e à toda a minha família. Vocês foram, são e serão sempre meu porto seguro e base para todas as minhas realizações.

A todos os meus amigos da faculdade e do CELD, tantos são que nem caberiam aqui, meu agradecimento por trazerem mais alegria e juventude aos meus dias.

À minha orientadora do projeto final, Silvana Rossetto, e ao meu orientador acadêmico, Luis Menasché. Obrigado por orientarem meus passos durante toda a minha jornada acadêmica, por sua constante solicitude e compreensão.

A todos vocês, meus eternos reconhecimento e gratidão.

RESUMO

SiMoni: um Sistema de Monitoramento de ambientes de trabalho

Hugo de Mello Dantas

Dezembro/2017

Orientador: Silvana Rossetto, D.Sc.

Desde a década de 40 do século XX, o estudo do ambiente de trabalho, das condições de trabalho mais adequadas e do emprego de tecnologias para alcançá-las revelou como a saúde do trabalhador pode ser profundamente afetada de acordo com as condições do seu ambiente de trabalho. Para atender às normas regulamentadoras, as condições de trabalho são aferidas manualmente por técnicos responsáveis e devem ser obtidas em cada local de trabalho, tarefa esta que pode ser lenta e desgastante caso existam muitos postos de trabalho a avaliar. O presente trabalho propõe um sistema automático de monitoramento das condições físicas de um ambiente de trabalho, tais como luminosidade e temperatura, visando garantir o cumprimento das Normas Regulamentadoras relativas à segurança e medicina do trabalho. Para tal, foram utilizadas tecnologias emergentes, tais como Redes de Sensores sem Fio e Internet das Coisas. O sistema proposto oferece ao usuário final uma aplicação WEB por meio da qual é possível monitorar de forma automática e remota as condições físicas de um determinado ambiente, sendo possível selecionar as grandezas físicas a serem observadas e os intervalos de observação das mesmas.

ABSTRACT

SiMoni: um Sistema de Monitoramento de ambientes de trabalho

Hugo de Mello Dantas

Dezembro/2017

Advisor: Silvana Rossetto, D.Sc.

Since the 1940s, the study of the work environment, its most suitable work conditions and the use of the best tecnologies to achieve them has revealed how the worker's health can be profoundly affected according the conditions of its own work environment. In order to accomplish the regulatory standards, some working conditions are manually checked by responsible technicians and must be obtained at each workplace, a task that can be slow and cumbersome if there are many jobs to evaluate. The present work proposes an automatic system to monitor the physical conditions of a work environment, such as luminosity and temperature, in order to guarantee compliance with the Norms Regulating the safety and occupational medicine. To do so, we used emerging technologies such as Wireless Sensor Networks and Internet of Things. The proposed system offers the end user a web application through which it is possible to automatically and remotely monitor the physical conditions of a given environment, being possible to configure the observed physical quantities and the observation intervals of the same.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

NR 17	Norma Regulamentadora 17
NR 09	Norma Regulamentadora 09
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
RSSF	Redes de Sensores Sem Fio
TCP	Transmission Control Protocol
RAM	Random Access Memory
RPL	Routing Protocol for Low Power and Lossy Networks
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
CoAP	Constrained Application Protocol
IETF	Internet Engineering Task Force
DODAG	Destination-Oriented Directed Acyclic Graphs
DIO	DODAG Information Object
DAO	Destination Advertisement Object
DIS	DODAG Information Solicitation

IEEE	Institute of Electrical and Electronics Engineers
MTU	Maximum Transmission Unit
SOAP	Simple Object Access Protocol
UDP	User Datagram Protocol
CoRE	Constrained RESTful Environments
SLIP	Serial Line Internet Protocol
USB	Universal Serial Bus
Ajax	Asynchronous JavaScript And XML
FIT	Future of Internet of Things
SSH	Secure Shell
JSON	Javascript Object Notation
URL	Uniform Resource Locator

Lista de Símbolos

bit	binary unit
kB	kilobytes

Sumário

Capítulo 1

Introdução

A ergonomia e a crescente preocupação em adaptar o ambiente de trabalho ao trabalhador ganhou significativo impulso a partir dos anos 1940, mais precisamente devido ao avanço da segunda guerra mundial. O sucesso das empreitadas bélicas começou a exigir um cuidado cada vez maior com fatores como desempenho e fadiga do militar que operava os equipamentos [39].

As normas de segurança do trabalho legislam sobre determinadas condições físicas, ergonômicas, biológicas e químicas que visam preservar os trabalhadores de efeitos adversos para a saúde decorrentes das suas condições de trabalho [7]. A NR 17[1], norma regulamentadora específica sobre ergonomia, dedica uma sessão inteira sobre os aspectos ideais do ambiente de trabalho. Destacam-se entre as condições ambientais a iluminação, a temperatura, o ruído, a umidade e a velocidade do ar. Auditorias são feitas com vistas a verificar se o ambiente atende às especificações e aos limites mínimo e máximo para cada uma dessas condições. As medidas são feitas em cada posto de trabalho.

A NR 17 não define propriamente com que frequência deve-se realizar auditorias por parte do empregador, porém, consta na NR-09[2] a necessidade de o empregador elaborar e implementar um Programa de Prevenção de Riscos Ambientais (PPRA) cuja avaliação deve ser efetuada sempre que necessário e pelo menos uma vez ao

[1] `http://trabalho.gov.br/images/Documentos/SST/NR/NR17.pdf`

[2] `http://trabalho.gov.br/images/Documentos/SST/NR/NR09/NR-09-2016.pdf`

ano. As normas dispostas na NR 17 devem, então, seguir essa recomendação, uma vez que o estudo da ergonomia também está relacionado à prevenção de incidentes no ambiente de trabalho pelo estudo do mesmo.

Uma vez que a NR 17 prescreve que sejam avaliados todos os postos de trabalho, a realização das medições das condições ambientais de trabalho pode ser uma tarefa exaustiva. A possível lentidão desse processo implicará em um atraso indesejável na tomada de ações para adequar o ambiente de trabalho às condições recomendadas pela norma, caso haja algum desvio.

1.1 Objetivos

O objetivo principal deste trabalho é propor uma solução automatizada para o monitoramento das grandezas físicas de um ambiente de trabalho, fazendo uso combinado de tecnologias emergentes tais como Redes de Sensores sem Fio e Internet das Coisas.

O objetivo secundário é que a arquitetura da solução proposta permita que a infraestrutura física de monitoramento possa ser compartilhada por agentes distintos (dono da empresa, órgão fiscalizador, sindicatos, etc.) para monitorar o ambiente de forma automática e remota.

1.2 Solução Proposta

Este trabalho propõe uma infraestrutura para monitoramento de ambientes de trabalho por meio de uma coleção de sensores cujo papel é coletar dados sobre grandezas físicas, tais como iluminação e temperatura. Estes sensores, por sua vez, estarão disponíveis por meio de seus endereços IPv6 [11], podendo ter suas informações acessadas através de simples requisições HTTP [14].

Note-se que esta solução é neutra em relação às aplicações finais, isto é, permite o desenvolvimento de aplicações independentes que fazem uso das informações coletadas. Como exemplo, é apresentado a SiMoni (Sistema de Monitoramento), uma

aplicação WEB que acessa os dados coletados e apresenta-os ao usuário em tempo real. Neste caso, o usuário poderia ser um técnico de segurança do trabalho que deseja redigir um relatório a ser enviado para o Ministério do Trabalho e que precisa medir a temperatura de cada posto de trabalho em tempo real e remotamente.

Assim, buscou-se atingir os objetivos supracitados utilizando-se uma rede de sensores sem fio capaz de realizar o monitoramento das condições do ambiente de trabalho e de disponibilizar o acesso a estes dados através de requisições HTTP, fazendo uso de protocolos e padrões da Internet.

1.3 Organização do texto

O restante deste texto está organizado em quatro capítulos. O segundo capítulo apresenta os conceitos básicos que fundamentam o desenvolvimento do trabalho mostrando um panorama geral de redes de sensores sem fio e os principais padrões utilizados na adaptação e integração dessas redes à Internet. O terceiro capítulo detalha a descrição do problema e mostra uma visão geral do sistema desenvolvido, discutindo seu projeto e sua implementação. O quarto capítulo mostra as avaliações realizadas no sistema, discutindo seus principais resultados. O último capítulo conclui o trabalho e apresenta possíveis direções para trabalhos futuros.

Capítulo 2

Conceitos básicos

Neste capítulo serão apresentados os principais conceitos que nortearam o desenvolvimento do sistema de monitoramento de grandezas físicas de um ambiente de trabalho, bem como a definição e as características dos protocolos utilizados para implementar a RSSF utilizada. Ao final do capítulo será dedicada uma sessão específica para apresentação de outros trabalhos na área que utilizaram os mesmos conceitos discutidos neste capítulo.

2.1 Redes de sensores sem fio e Internet das Coisas

Durante a década de 1990, observou-se avanços na área de microprocessadores, em particular sistemas micro-eletro-mecânicos [24]. Esses avanços possibilitaram o desenvolvimento de *smart sensors*, ou sensores inteligentes [41], os quais são dotados de capacidade de processamento, armazenamento de dados, comunicação sem fio e sensoreamento de grandezas físicas. Normalmente, busca-se utilizar esses sensores em larga escala, como no monitoramento de uma floresta. Para atingir este objetivo, sem perda da precisão da medida que se queira realizar, tais sensores costumam ter capacidade computacional limitada e tamanho reduzido, o que barateia seu custo.

Esses sensores inteligentes, trabalhando cooperativamente e trocando informações entre si, compõem o que se conhece como Redes de Sensores Sem Fio (RSSF). O

modo de operação das RSSFs é ditado pelas características que estas redes apresentam. Dada a limitação da energia disponível, é comum que os sensores não fiquem sempre ativos. Da mesma forma, algoritmos de roteamento e protocolos devem ser projetados visando otimizar o consumo de energia. Daí a importância de dotar essas redes de capacidade de auto-organização, para que a entrada ou a saída de um sensor da rede não interrompa seu funcionamento.

As RSSFs também podem rotear os dados coletados até a estação base de diferentes formas. Uma estação base neste contexto representa um polo principal responsável pela cobertura de uma determinada área da rede. Como a transmissão de dados é a operação que mais demanda energia no contexto de RSSF [12], diferentes formas de roteamento entre um nó e sua estação base foram propostas, visando uma utilização mais eficiente em termos de consumo de energia, considerando a natureza da aplicação.

Uma primeira estratégia de roteamento seria a transmissão direta do sensor à estação base, vantajosa quando a comunicação envolve distâncias pequenas (menores do que 30 metros). Porém, distâncias maiores podem ser alcançadas utilizando-se os próprios sensores como nós intermediários da transmissão. De acordo com o tipo de aplicação que se deseja desenvolver, esse roteamento envolvendo nós intermediários trás uma dupla vantagem. Primeiro, a possibilidade de se preprocessar estes dados dentro da própria rede, visando melhorar sua precisão, como por exemplo a diminuição de ruídos. Segundo, a capacidade de agregação e fusão dos dados obtidos, obtendo economia do consumo de energia pela diminuição do volume de dados. [24]

Ainda no contexto das RSSFs, o endereçamento dos dispositivos conectados é uma questão que merece alguma atenção. De acordo com a finalidade a que os dispositivos em rede se objetivem, os sensores podem ser endereçados unicamente ou não.

Dada a natureza particular das RSSFs, inicialmente para cada aplicação desenvolvia-se um algoritmo de roteamento próprio, diretamente integrado à aplicação. A integração com outras redes era feita por meio de *gateways* especiais, responsáveis por fazer a conexão da RSSF com a Internet.

Esta integração, contudo, apresenta alguns desafios, muitos justificados pelo baixo poder computacional dos dispositivos ligados às RSSFs. Historicamente, o desenvolvimento da Internet acompanhou o desenvolvimento do poder computacional dos computadores ao longo dos anos 80 e 90. A própria pilha TCP/IP, hoje padronizada na Internet, não é possível de ser implementada nos dispositivos que se espera encontrar normalmente em RSSFs, pois estes não possuem poder de processamento e nem memória de armazenamento suficientes.

A exemplo do que ocorreu com a Internet, quando o protocolo IP começou a ser mais amplamente utilizado, diversas redes privativas que não implementavam este protocolo viram-se inicialmente obrigadas a implementar *gateways* para garantir a interoperabilidade com as demais redes. Porém, os *gateways* foram rapidamente preteridos uma vez que sua adoção traz problemas como o aumento da complexidade da arquitetura de rede e a falta de flexibilidade e escalabilidade. Uma situação análoga pode ser observada na integração das RSSFs e a Internet. A adoção fim-a-fim do protocolo IP nessas redes facilita em muito a implantação dessas redes, trazendo vantagens como interoperabilidade, flexibilidade, versatilidade e escalabilidade. Em razão disso, mais recentemente, foram desenvolvidas versões compactadas da pilha de protocolos da Internet para implementação nas RSSFs, e as RSSFs passaram a integrar o que hoje é chamado de "Internet das Coisas"(IoT).

Graças a esses esforços, as RSSFs de uma forma geral podem hoje operar sem a utilização de *gateways* especiais, sendo diretamente integradas à Internet. O endereçamento dos dispositivos utilizando IPv6 é facilitado graças ao suporte que os sistemas operacionais específicos para sistemas embarcados oferecem, como é o caso do sistema operacional Contiki, utilizado no presente trabalho.

No paradigma da Internet das Coisas (do inglês *Internet of Things*) (IoT) [16] os objetos com os quais interagimos diariamente estarão conectados a uma rede, trocando dados em tempo real com o objetivo de oferecer novas experiências personalizadas ao usuário, adicionando inteligência a estes objetos.

Uma pessoa, após um exaustivo dia de trabalho, poderia, antes mesmo de chegar em sua residência, regular a temperatura do ar-condicionado de sua sala de estar, ao

mesmo tempo em que consultaria a sua geladeira para conferir que alguns ingredientes já foram utilizados. À porta de casa, seria possível verificar que todas as luzes da casa foram apagadas uma hora atrás, indicando que seus familiares já estariam possivelmente dormindo, de onde se conclui que nosso personagem possivelmente se atrasou para o jantar.

A grande adesão das pessoas do mundo inteiro ao acesso à Internet através de dispositivos móveis pode ser considerado um passo na direção deste paradigma, uma vez que dessa forma a Internet está acessível em qualquer lugar.

O sensoriamento automatizado de um determinado ambiente, com o objetivo de monitorar certas condições como luminosidade e temperatura, é um exemplo de aplicação de IoT que pode ser utilizado em diferentes contextos.

2.2 Protocolos da Internet para RSSF

Esta seção tem por objetivo apresentar os protocolos utilizados para integrar as RSSF à Internet, bem como descrever em linhas gerais o funcionamento específico de cada um.

2.2.1 6LoWPAN

Embora muito vantajosa, a adoção dos protocolos da Intenet por dispositivos embarcados não é uma tarefa simples uma vez que estes protocolos não são otimizados para operarem em dispositivos com capacidades computacionais restritas. Dentre os desafios que a adoção do protocolo IP traz pode-se destacar problemas relacionados à segurança, gerenciamento e tamanho dos pacotes de dados [31].

Para se utilizar o protocolo IPv6 em redes IEEE 802.15.4, padrão da camada de enlace utilizado em redes sem fio de baixa taxa de transmissão, é necessário adequar o MTU de 1280 bytes do IPv6 ao MTU de 127 bytes suportado pelo padrão IEEE 802.15.4 [9]. Visando a superação desse desafio, o IETF (*Internet Engineering Task Force*), grupo responsável pela criação e manutenção dos padrões adotados na In-

ternet, designou um grupo, o 6LoWPAN, especialmente para projetar uma solução adequada para essa questão. Este grupo propôs um padrão com o mesmo nome do grupo (6LoWPAN [23]) cuja principal funcionalidade consiste em uma camada de adaptação entre a camada de rede e de enlace. A esta camada cabe realizar a compressão do cabeçalho IPv6, a fragmentação e a reconstrução dos pacotes transmitidos e recebidos das camadas inferiores [25]. A compressão do cabeçalho IPv6 está ilustrada na Figura 2.1, realizada principalmente através da remoção de alguns campos, como versão e classe de tráfego, e da diminuição de campos como os endereços de origem e destino pela inferência dos mesmos pelos endereços da camada de enlace e pelo prefixo de rede local [18].

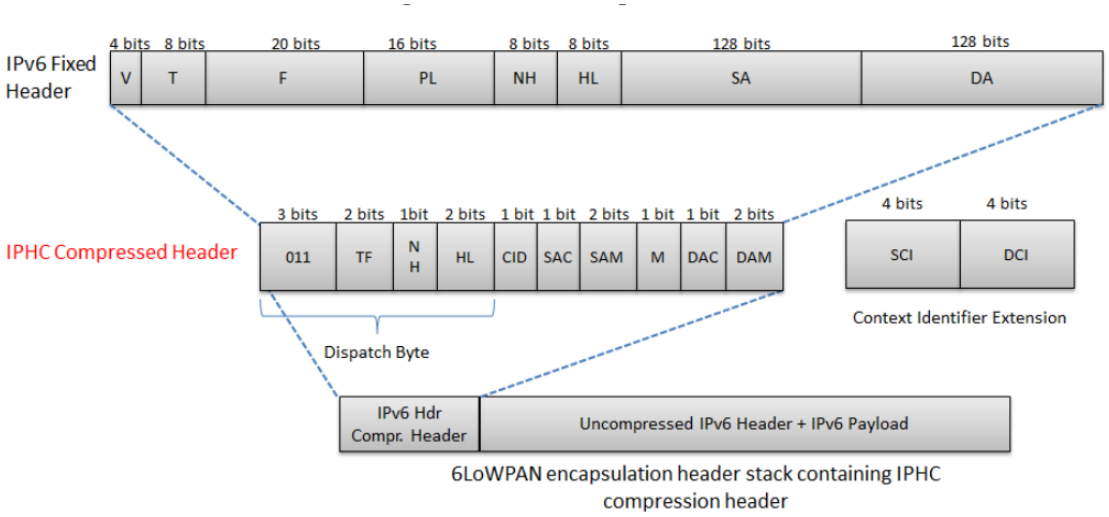

Figura 2.1: Compressão do cabeçalho IPv6 [18]

2.2.2 RPL

Como já mencionado, as RSSFs — e mais especificamente as redes de baixa potência com perdas (*Low Power and Lossy Networks (LLN)*) — trazem diversos desafios para sua implementação, muitos deles justificados pelo baixo poder computacional dos dispositivos que as constituem. Diante disto, o IETF propôs um protocolo de roteamento voltado para redes 6LoWPAN, denominado RPL[40], cujo objetivo é tratar de forma mais adequada as especificidades dessas redes e minimizar o tráfego de controle [4].

O RPL é um protocolo de roteamento que implementa um vetor de distâncias e faz uso do protocolo IPv6. Para lidar com o modo de operação essencialmente dinâmico das RSSFs, o RPL combina diferentes estratégias de roteamento, o que o torna altamente flexível. Isto porque é esperado que a RSSF esteja preparada para lidar com a heterogeneidade dos dispositivos que formam a rede, com a entrada e saída de novos dispositivos, assim como com a interrupção do funcionamento de algum dispositivo ao longo da execução de uma aplicação.

O RPL define suas rotas formando uma árvore de roteamento [19]. O protocolo trata os dispositivos como nós, formando com eles grafos acíclicos direcionados. Uma vez que as arestas informam a direção dos nós destinos, é dito que o RPL forma um ou mais DODAG (grafo acíclico direcionado orientado a destino). Para a formação deste grafo, o nó raiz emite para os vizinhos uma mensagem (do tipo DIO) que informa aos mesmos o *rank* e a função objetivo do nó raiz.

É na função objetivo que se define a métrica segundo a qual os nós selecionam seus pais, possibilitando que a rede utilize rotas que contemplem interesses específicos de cada aplicação. O projeto da função objetivo também deve levar em conta a definição da forma como deve ser calculando o *rank*, que define o valor que representa cada nó e que permite ao protocolo evitar a ocorrência de *loops* no DODAG.[34]

A partir daí cada nó calcula seu *rank* na rede e emite novas mensagens DIO para seus próprios vizinhos. Quando todos os nós tiverem recebido mensagens DIO estarão definidas as rotas ascendentes, isto é, rotas que levam do nó raiz até todos os demais nós da rede. Para a formação de rotas no sentido inverso, um processo semelhante é deflagrado, porém através do envio de mensagens DAO destinadas ao nó raiz.

Assim, o RPL adota uma solução baseada em IP para que o acesso aos sensores de uma rede sem fio possa ser feito sem a utilização de *gateways* [38]. Para compôr a rede, além dos sensores que monitoram as condições do ambiente, é necessário que exista um sensor específico para atuar como um roteador de borda. De certa forma, o roteador de borda faz as vezes de um *gateway* entre a Internet e a RSSF, na medida que ele articula a relação entre as duas.

Todavia, seria um *gateway* mínimo que simplifica a infraestrutura da rede, se comparado com a utilização de um *gateway* padrão. Isso porque não é necessário um equipamento com configurações específicas, pois utiliza-se um dos nós da rede para fazer o roteamento externo da própria rede, aumentando a integração entre as redes interna e externa pela utilização de um mesmo protocolo.

Além disso, o próprio roteador pode ser acessado através de um endereço IPv6, diminuindo a sua complexidade de funcionamento em relação a um *gateway* padrão. Dessa forma, os dados podem ser acessados através da Internet através de endereços IPv6 que identificam unicamente cada sensor disponível na rede.

Para a rede de sensores, o sensor específico utilizado como roteador de borda será o nó raiz do grafo DODAG montado pelo RPL. Para a rede externa, este sensor irá atribuir um endereço IPv6 para cada um dos sensores disponíveis e através dele irão passar todas as solicitações.

2.2.3 CoAP

O CoAP [32] é um protocolo da camada de aplicação projetado especificamente para dispositivos de baixo poder computacional. Ele traz como principal vantagem a possibilidade de acesso a recursos de um dispositivo através de serviços RESTful. Em outras palavras, é possível a interação com os recursos do dispositivo a partir de métodos bem conhecidos e padronizados como o *GET*, *POST*, *PUT* e *DELETE*.

Em redes LLN, a utilização do protocolo HTTP é problemática devido a uma série de razões, entre elas o fato do HTTP usar o TCP na camada de transporte, cuja robustez ultrapassa as possibilidades computacionais da rede. Além disso, o pequeno tamanho limite de quadros da camada de rede traria uma maior fragmentação de pacotes IP, aumentando a quantidade de processamento para a fragmentação e reorganização dos pacotes. Para lidar com essas dificuldades, o CoAP adota os conceitos de "transferência em blocos"(*block-wise transfer*), isto é, direciona para a aplicação a responsabilidade de fragmentar e recuperar os blocos de informação, respeitando o limite máximo imposto pelas camadas mais baixas da pilha de protocolos. O CoAP utiliza o protocolo UDP [29] e assume a responsabilidade de garantir

a entrega de pacotes, caso o desenvolvedor solicite este serviço.

No entanto, o CoAP não se limita a ser uma versão resumida do HTTP, mas ao contrário, ele amplia a sua capacidade de operação, trazendo funcionalidades especialmente interessantes para aplicações no contexto de IoT. As funcionalidades `observe` e `discovery` destacam-se neste quesito. `observe` implementa a abstração de publicação e assinatura (*publish/subscribe*). O cliente, quando deseja monitorar um determinado recurso, informa seu desejo na requisição *GET*. O servidor, então, se encarregará de enviar uma resposta CoAP padrão para o cliente cada vez que o estado deste recurso sofrer alguma alteração. Por sua vez, o `discovery` traz a informação dos recursos disponíveis na plataforma. Este método pode ser utilizado tanto para o descobrimento de serviços quanto para o descobrimento de recursos. A resposta CoAP para uma requisição `discovery` retorna diversos metadados sobre os recursos disponíveis.

2.2.4 REST

O padrão REST [15] caminha na mesma direção dos esforços da Internet das Coisas uma vez que permite acesso aos recursos da rede através de métodos do protocolo HTTP (*GET*, *POST*, *PUT* e *DELETE*) e torna desnecessário o emprego de arquivos extras e de *parsers* — comumente empregados em protocolos como o SOAP — diminuindo a complexidade de operação da rede.

Uma vantagem interessante do uso do padrão REST é que este padrão pressupõe a criação de recursos visando a reutilização dos mesmos [17]. No contexto de Internet das Coisas, é interessante que os objetos inteligentes ofereçam serviços através deste padrão, pois isto possibilita o desenvolvimento de uma infraestrutura de rede capaz de atender as demandas de diferentes aplicações.

Neste trabalho, o padrão REST foi empregado utilizando o protocolo CoAP, um protocolo que pode ser visto como uma alternativa ao HTTP para o contexto de dispositivos com restrições de processamento.

2.3 O sistema operacional Contiki

O desenvolvimento de sistemas operacionais capazes de serem executados em sensores integrantes de RSSFs não é uma tarefa trivial. Em um ambiente típico de sensoriamento, espera-se que estes dispositivos utilizem microcontroladores de baixa capacidade de processamento, memória de código da ordem de 100kB e memória RAM inferior a 20kB.[13]

Contiki [13] é um sistema operacional para dispositivos embarcados especialmente projetado para o desenvolvimento de aplicações para RSSFs com baixa taxa de transmissão e baixo consumo de energia. Uma de suas principais características é oferecer um modelo de programação orientado a eventos, como uma solução para permitir concorrência entre os processos compartilhando a mesma pilha de memória. Contudo, programas que exigem um tempo maior de processamento monopolizariam a CPU no processamento orientado a eventos, uma vez que, iniciado um evento, este deverá ser processado até o seu término. Reconhecendo estas dificuldades, o Contiki permite opcionalmente a utilização de *multi-threading* com preempção através de bibliotecas explicitamente chamadas pelo programador. A biblioteca de *multi-threading* é dividida em duas partes, uma independente da plataforma onde o Contiki esteja sendo executado, e uma específica para esta. A API da biblioteca traz funções que permitem ao programador o controle da execução de cada *thread*. Por combinar o uso de *multi-threading* com programação orientada a eventos, pode-se dizer que o Contiki oferece um modelo híbrido de programação.

Uma outra característica desse sistema operacional é permitir a programação dinâmica de uma rede de sensores. Isso é particularmente útil para projetos em que seja necessário corrigir erros nos códigos implementados, mas inviável a interrupção da atividade da rede para a manutenção, como no caso de sensores localizados em zonas de difícil acesso. Para isso, os desenvolvedores implementaram um protocolo simples, que consiste em transmitir o novo código para um nó receptor, e da sucessiva transmissão para os nós vizinhos através de *broadcasts*. Testes mostraram que a reprogramação de uma imagem de 30 kB demorou em média 30 segundos. [13]

O uso do Contiki neste trabalho trouxe como grande vantagem o fato deste sistema já oferecer suporte aos principais protocolos de integração das RSSFs com a Internet (6LoWPAN, RPL e CoAP), os quais são adotados no desenvolvimento deste trabalho.

2.4 Trabalhos relacionados

Diversos trabalhos fizeram uso dos mesmos padrões e protocolos considerados neste capítulo, como a utilização de dispositivos implementando o padrão IEEE 802.15.4, o padrão 6LoWPAN, o acesso aos dispositivos através de *web services* utilizando o protocolo CoAP em conjunto com endereçamento IPv6 e o protocolo RPL. As aplicações envolvem a implementação de *healthcares*, automação residencial e sensoriamento remoto de ambientes.

Em [1] os autores propõem a utilização de sensores para o monitoramento de um estacionamento. Para cada vaga, disponibiliza-se um sensor. O motorista poderá acessar o sistema através de um endereço web e obter informações como a quantidade de vagas disponíveis. Para solicitar uma vaga, o motorista informará os dados relativos ao seu veículo e o tempo que pretende manter o mesmo estacionado, e o sistema reservará uma vaga baseado nas informações enviadas pelo usuário. Após o esgotamento do tempo, o sistema tratará de liberar a respectiva vaga.

Em [27] propõe-se a implementação de um *healthcare*, onde cada paciente contará um conjunto de sensores para o monitoramento suas necessidades médicas. A equipe médica, neste caso, poderá acessar cada sensor através de uma URL específica. Utiliza-se o recurso *discovery* do protocolo CoAP para obter os recursos disponíveis no sensor acessado. Este trabalho também considera a utilização do algoritmo de chave pública utilizando curvas elípticas para a garantia da segurança da informação do sistema.

Em [5] os autores apresentam a implementação de uma plataforma de gerenciamento de energia residencial. Para isto, utiliza-se sensores cuja função é medir parâmetros importantes para aferir o consumo de energia da casa, como a corrente e

a voltagem, e informar ao usuário através de uma interface. Em conjunto com este sistema, propõe-se a execução de um algoritmo de otimização, visando otimizar o consumo de energia mensurado pelo sistema.

Assim como este trabalho, [1] e [27] apresentam resultados a partir de simulações utilizando o simulador Cooja (apresentado na sessão 4.1.1). Entretanto, os trabalhos citados não implementam uma interface própria para interação com o usuário. [5] utiliza a aplicação web *Emoncs*[1], enquanto [27] e [1] propõem que a interação com os servidores CoAP em execução nos sensores seja realizada utilizando o Copper [20], um navegador web específico para Internet das Coisas.

[1] `https://emoncms.org/`

Capítulo 3

Problema e solução proposta

Este capítulo tratará do problema que este trabalho se propôs a solucionar, bem como da solução desenvolvida. Serão descritos a arquitetura da solução, a sua implementação, as ferramentas e bibliotecas utilizadas, e por fim o funcionamento da aplicação web que consome as informações disponibilizadas pela RSSFs implementada. Todo o código fonte da aplicação SiMoni e dos programas executados nos sensores encontram-se no seu repositório [10].

3.1 Descrição do problema

Há no Brasil normas regulamentadoras do trabalho que prescrevem condições ideais do ambiente de trabalho visando o conforto e principalmente a saúde do trabalhador. A ergonomia é a ciência que se preocupa com o estudo da melhoria das condições de trabalho por meio de sistemas, máquinas e equipamentos [3].

Existem diversos fatores que influenciam na qualidade e eficiência do trabalho, bem como no conforto e na saúde do trabalhador, dentre eles, fatores ambientais como iluminação, quantidade de ruído, temperatura e umidade do ar. Estes fatores são considerados nas normas regulamentadoras, em especial na NR 17. A partir de determinados limites podem ocorrer problemas como dificuldade na comunicação, no sono, surgimento de estresse (*stress*), bem como dificuldades de ordem mental

e/ou emocional [36].

Assim, profissionais de segurança do trabalho atuam no monitoramento dessas condições e em medidas capazes de garantir o cumprimento das mesmas, sendo também responsáveis pela elaboração de relatórios que o comprove. Porém, uma vez que é preciso que o monitoramento seja feito em cada posto de trabalho, é comum gastar-se algum tempo com os equipamentos de medida em mãos registrando local a local a grandeza desejada, como quantidade de luz e temperatura. Uma aplicação web que mostrasse estas leituras em tempo real facilitaria este esforço, uma vez que não só seria possível obter as leituras de múltiplas estações de trabalho de forma simultânea, como também diminuiria a necessidade da presença física do profissional responsável pela medida.

Para desenvolver uma plataforma que auxiliasse técnicos e profissionais de segurança do trabalho a monitorarem as condições de um ambiente de trabalho, buscou-se a utilização de tecnologias e padrões emergentes associados às RSSFs e Internet das Coisas. A solução proposta visa permitir que uma mesma infraestrutura física de monitoramento, formada por uma rede de sensores sem fio, possa ser utilizada por diferentes aplicações fazendo uso de protocolos da Internet.

Na atual etapa de desenvolvimento da computação ubíqua e da própria Internet das Coisas, a busca por soluções padronizadas multiplica as possibilidades de utilização da tecnologia, uma vez que os desenvolvedores podem direcionar seus esforços nas aplicações em si, investindo menos tempo em desafios como lidar com heterogeneidade da rede, gerenciamento de *gateways*, entre outros. Seguindo este pensamento, é altamente vantajoso o desenvolvimento de aplicações para objetos inteligentes que disponibilizem os recursos que estes objetos oferecem por meio de interfaces de uso genérico. Dessa forma, os recursos estarão acessíveis de forma independente. Nessa direção, a aplicação desenvolvida neste trabalho disponibiliza o acesso à leitura dos sensores da infraestrutura de monitoramento de um ambiente de trabalho por meio de uma API RESTful executada em um servidor CoAP mínimo.

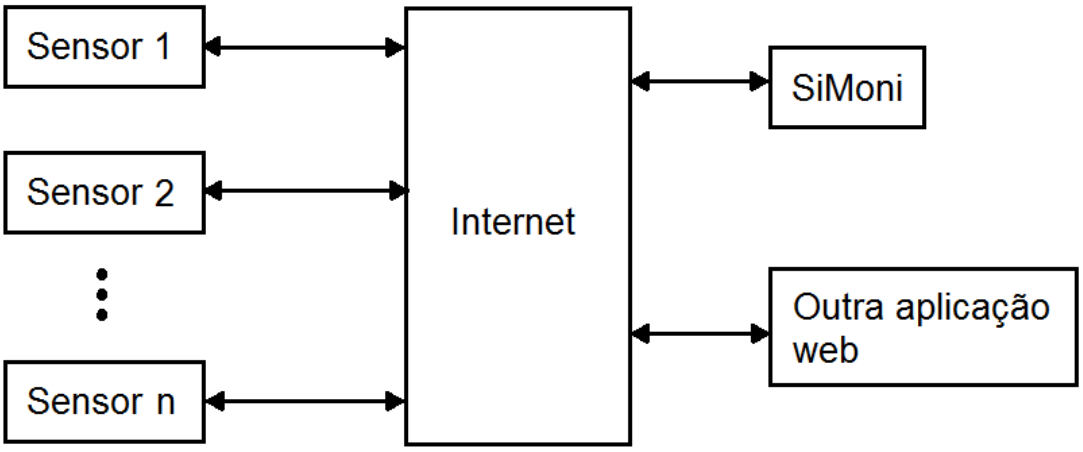

Figura 3.1: Diagrama em blocos da solução proposta.

3.2 Descrição da solução proposta

A solução proposta é constituída de um sistema integrado, composto fundamentalmente por uma rede de sensores e por uma aplicação web.

A Figura 3.1 mostra esses blocos fundamentais. Estão representados nesta figura diferentes aplicações acessando as informações disponibilizadas pelos sensores através de serviços web, uma vez que a infraestrutura da rede foi implementada de maneira independente da aplicação final, possibilitando a utilização da mesma rede para diversas aplicações finais.

3.2.1 Casos de uso

O diagrama de caso de uso da Figura 3.2 destaca as possíveis utilizações do sistema.

As duas principais utilizações do sistema constituem em obter as medidas enviadas pelos sensores em tempo real, salvar estes dados e buscar por dados salvos anteriormente. A pesquisa do histórico de informações salvas pode ser feita opcionalmente através da pesquisa por uma data em que a observação tenha sido registrado. Por "observação", deve-se entender o nome correspondente ao conjunto de dados salvos no banco de dados. Já a leitura em tempo real é obtida com a seleção da

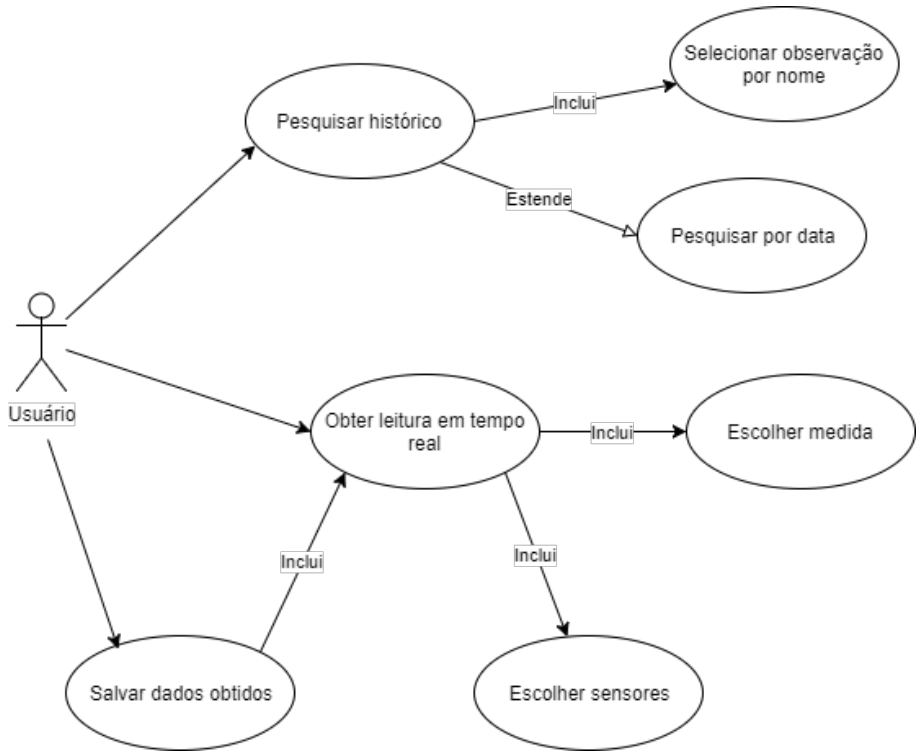

Figura 3.2: Casos de uso do sistema.

grandeza física que se deseja monitorar e dos sensores que se deseja consultar. Após interromper a leitura, o usuário poderá salvar estes dados, informando um nome para indicar a observação.

3.2.2 Projeto da solução

A rede de sensores é constituída por um conjunto de dispositivos dotados de comunicadores via rádio. Os sensores possuem um raio de abrangência limitado de acordo com cada modelo. Os sensores utilizados neste trabalho possuem um raio de aproximadamente 50 metros.

A solução consiste na utilização de sensores programados para coletar medidas das condições ambientais ao seu redor. Como, por uma indicação expressa da NR 17, deve-se realizar medidas de temperatura, luminosidade, umidade, pressão e ruído em cada local de trabalho, é neste local que deverá estar localizado cada um dos sensores. Estes sensores, entretanto, não funcionam apenas como coletores daquelas informações, mas também atuam como roteadores da RSSF.

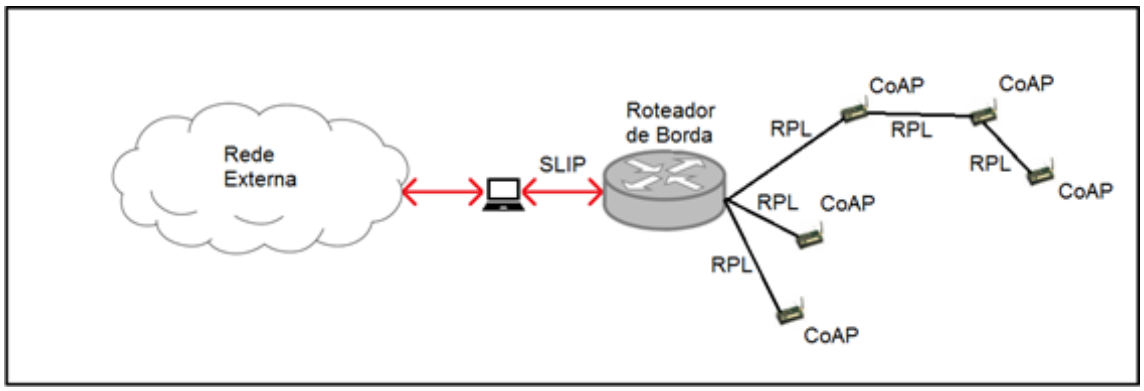

Figura 3.3: Arquitetura da solução.

A Figura 3.3 ilustra a arquitetura geral da solução proposta. Segundo esta arquitetura, a rede externa e a RSSF comunicam-se através roteador de borda, que é também um sensor executando uma aplicação específica para o roteamento da RSSF rodando sobre o sistema operacional Contiki. Este roteador de borda deve estar conectado a um PC conectado à Internet através de uma conexão USB.

Para permitir o tráfego IP através dessa conexão USB, o Contiki dispõe de um utilitário chamado *Tunslip*. A versão apta a trabalhar com IPv6 é chamada *Tunslip6*. O *Tunslip* cria uma conexão através do protocolo SLIP [30], um protocolo antigo porém capaz de estabelecer uma conexão serial com menor *overhead* comparando-se com protocolos mais utilizados atualmente como o PPP [33]. Essa conexão SLIP é habilitada através de uma interface *TUN*, que simula um device da camada de rede (camada 2) da pilha TCP/IP. Com isso, o computador e o roteador de borda se conectam logicamente e assim a RSSF passa a estar acessível a requisições externas.

3.2.3 Implementação da solução

Para implementar a camada de roteamento da rede utilizamos o protocolo RPL. Graças a ele é possível garantir rotas a todos os sensores, desde que cada sensor seja vizinho de pelo menos um outro sensor.

Em cada um dos sensores é carregado um código responsável por executar um servidor CoAP no próprio dispositivo. Este servidor utiliza um serviço RESTful para disponibilizar acesso aos recursos do dispositivo, que no caso são as grandezas

físicas que o sensor é capaz de medir e que seja de interesse da ergonomia, a saber, temperatura, luminosidade, velocidade do ar, umidade e níveis de ruído. O servidor oferece também a funcionalidade de descoberta de serviços, em que o servidor responde com os serviços disponíveis no momento.

No roteador de borda, um servidor HTTP é executado cujo único serviço disponível é um recurso que retorna a lista de endereços dos sensores disponíveis. Embora este protocolo não seja o mais adequado para funcionar em sensores de baixa capacidade de processamento, a sua utilização não sobrecarrega o processamento do sensor nem a capacidade de transmissão da rede pois espera-se que este serviço, que responde apenas à requisições *GET*, seja pouco requisitado, já que após obter a lista dos endereços dos sensores, o usuário poderá interagir diretamente com cada um destes.

A escolha pelo HTTP também torna mais fácil a interação com este nó, que é fundamental para o funcionamento do restante do sistema. Este é o único sensor que precisa estar ligado a um computador, através de uma conexão SLIP, que permite conectar a rede de sensores ao mesmo através de uma conexão USB.

3.2.4 Visão geral do funcionamento da aplicação SiMoni

Primeiramente, todos os sensores são inicializados. Dentre os sensores da rede, um deles deve executar o código responsável por torná-lo o roteador de borda da rede. Os demais sensores executam o código de um servidor web que oferece acesso aos recursos do próprio sensor. O Contiki possui mecanismos e oferece suporte a protocolos capazes de identificar e mapear quais sensores são alcançáveis para o roteador de borda.

Quando o usuário abre a aplicação web, o cliente envia uma requisição Ajax [28] para o servidor solicitando a lista de sensores disponíveis. O *backend*, por sua vez, realiza uma requisição HTTP para o roteador de borda, que mantém uma estrutura de dados com os endereços e rotas dos sensores disponíveis na rede. Observa-se que essa lista é dinâmica e recebe atualizações periódicas dos estados dos sensores. A figura 3.4 mostra a tela inicial da aplicação.

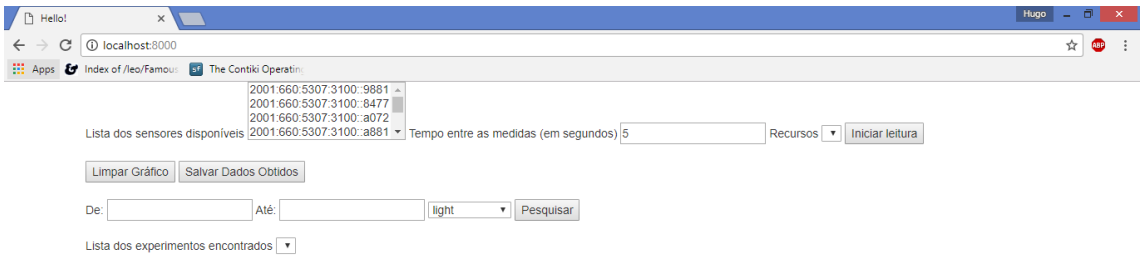

Figura 3.4: Visão da tela inicial da aplicação.

3.2.4.1 Leitura em tempo real

O usuário escolhe da lista de sensores aqueles a partir dos quais deseja obter leituras em tempo real. O *backend*, então, cuida de gerar uma lista de recursos comuns a todos os sensores que o usuário marcou. Interessante observar que estes recursos são obtidos através do recurso `discovery`, que lista quais recursos a API RESTful do sensor está disponibilizando. O usuário terá a visualização dos dados requisitados conforme mostrado na Figura 3.5.

A partir daí, pode-se selecionar um recurso disponível e um ou mais sensores para que se obtenha uma leitura em tempo real, obtida segundo uma frequência também informada pelo usuário. O sistema começa, então, a realizar uma série de requisições *POST* para o servidor, enviando o endereço do sensor específico e o recurso escolhido. O servidor, por sua vez, dispara uma requisição CoAP diretamente para o sensor para colher o dado solicitado. Na interface do usuário, um gráfico será renderizado dinamicamente conforme os dados forem sendo retornados. A Figura 3.6 ilustra o diagrama de sequência relativo às etapas necessárias para exibição de dados para o usuário.

Uma vez que o usuário interrompa o processo de leitura em tempo real, é possível

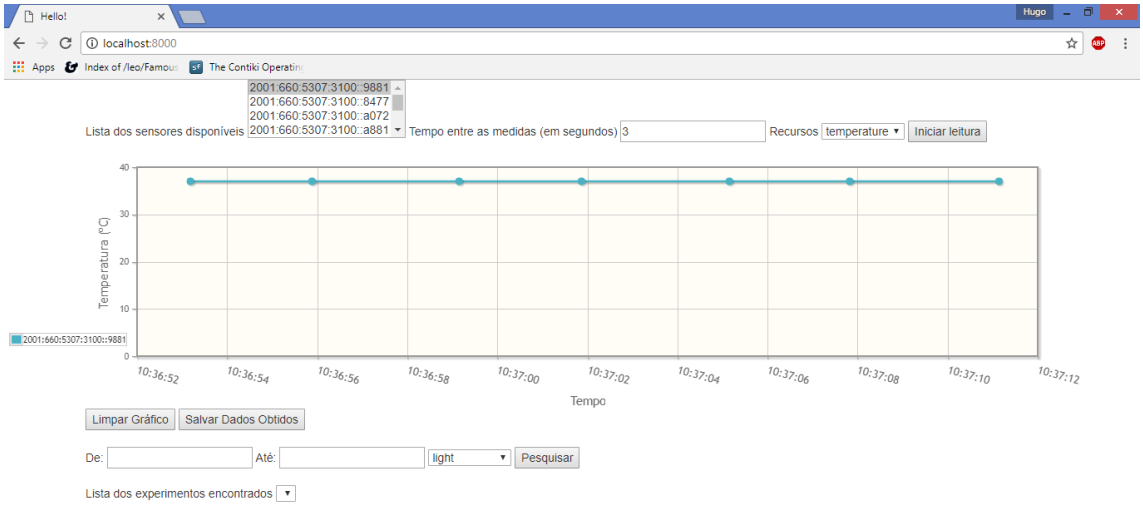

Figura 3.5: Visão da aplicação renderizando o gráfico com as informações provenientes do sensor escolhido.

salvar os dados obtidos, informando um nome para a série de dados recebidos dos sensores. É importante observar que o sistema não permite que um mesmo nome seja utilizado para coletas diferentes, visando otimizar a pesquisa por estes dados que o usuário possa vir a realizar posteriormente. A Figura 3.7 ilustra esta etapa.

3.2.4.2 Pesquisa de dados armazenados

Um usuário poderá também pesquisar por medidas que tenham sido salvas na base de dados. Para isso, deverá informar obrigatoriamente qual grandeza deseja pesquisar (luminosidade, temperatura, pressão e umidade) e opcionalmente uma data de início e uma data de fim. As coletas encontradas serão listadas por nome, e ao serem selecionadas, o gráfico mostrando os respectivos dados será apresentado na interface para o usuário. Esta operação está ilustrada no diagrama de sequência da Figura 3.8. A Figura 3.9 ilustra uma pesquisa feita por uma observação de um sensor de luminosidade.

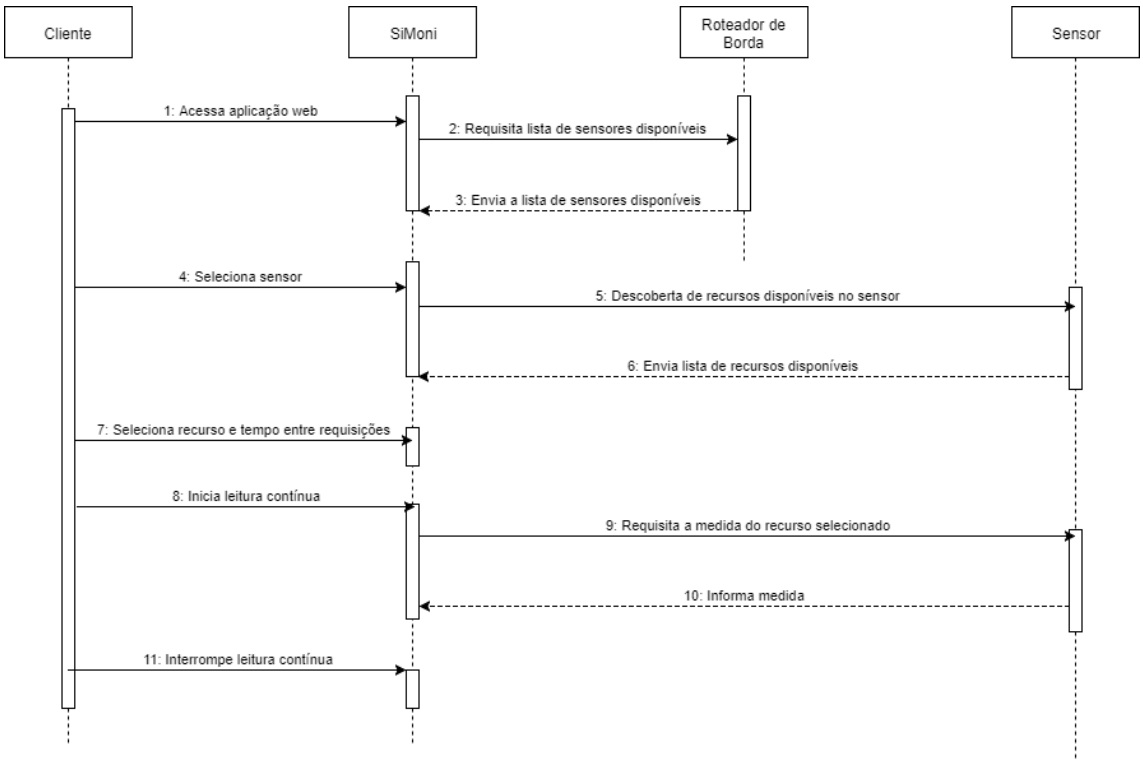

Figura 3.6: Diagrama de sequência relativo à obtenção de dados em tempo real.

3.2.5 Dificuldades na implementação da solução proposta

A solução proposta prevê que os sensores disponibilizem a leitura das medidas das condições ambiente de uma forma genérica o suficiente para poderem ser utilizadas por qualquer aplicação que deseje essas informações, e que dará por sua vez um objetivo a essas medidas.

No caso deste trabalho, a aplicação web é totalmente focada para o uso dos recursos do sensores como fontes de informação para a área de segurança do trabalho. Algumas dificuldades, entretanto, impediram que a solução fosse plenamente implementada.

Uma das dificuldades encontrada foi o fato de não se ter encontrado um hardware que conseguisse monitorar todas as grandezas destacadas pela NR 17. O escopo, então, foi reduzido ao monitoramento de temperatura e luminosidade, por serem essas grandezas mais facilmente encontradas nos hardwares empregados pelas plataformas onde este trabalho foi desenvolvido.

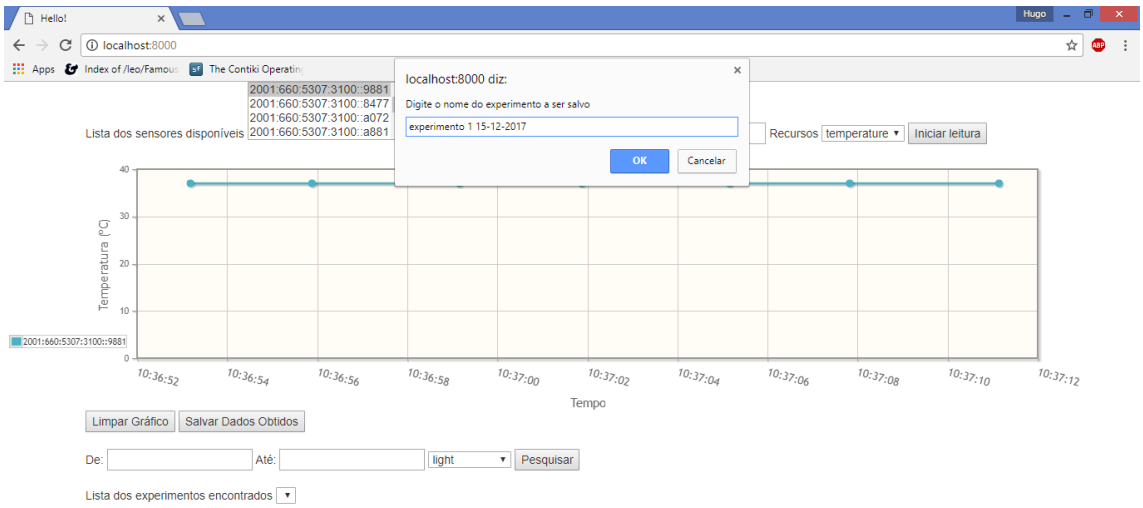

Figura 3.7: Visão da etapa de registro dos dados observados.

Uma segunda dificuldade foi com relação à quantidade de informações que cada sensor é capaz de enviar. Considere-se que não basta que o sensor envie apenas o valor numérico da medida coletada; outras informações são necessárias, como a unidade dessa medida. A dificuldade está no fato de que, pelas limitações naturais do projeto discutidas no capítulo 2, a quantidade de informações enviadas a cada requisição pelos sensores deve ser a mínima possível.

Este trabalho operou com blocos de mensagens sendo enviado pelos sensores menores ou iguais a 128 *bytes*, pré-definido na implementação do protocolo CoAP utilizada no Contiki. Segundo [21], pacotes maiores do que 128 bytes limitariam o tamanho do *buffer* de IP e outros recursos importantes para a manipulação de retransmissões.

Os dados obtidos pelo sensor são enviados no formato *string*. Como cada caractere ocupa um espaço de 1 *byte*, em termos práticos, os sensores enviam *strings* de até 128 caracteres. Como era preciso enviar os dados no formato JSON (discutido na seção 3.2.6), todas as mensagens no formato JSON precisam ser convertidas em *string*, o que traz limitações para a quantidade de informação enviada. Por exemplo, enquanto é possível representar um número com seis casas decimais através do tipo *float*, ocupando apenas 4 bytes, o mesmo número convertido em *string* ocuparia 6

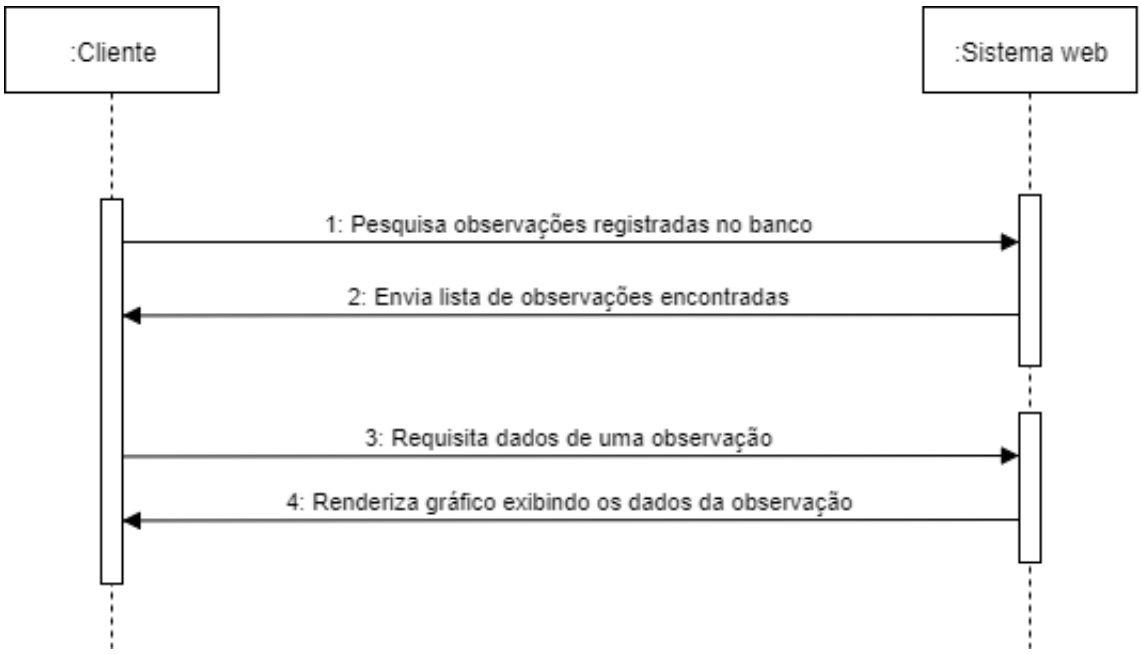

Figura 3.8: Diagrama de sequência relativo à pesquisa de histórico.

ou 7 caracteres, e portanto 6 ou 7 bytes.

A Figura 3.10 mostra a quantidade de bytes utilizados para a representação de cada informação. Considere-se que cada caractere ocupa 1 byte dos 128 disponíveis para a composição da mensagem. Considerou-se no cálculo os caracteres necessários para a identificação de cada campo, ou seja, os atributos da mensagem JSON. No gráfico, deve-se entender por "dado"a medida realizada pelo sensor; "idSensor"como o identificador do sensor responsável pela medida; "iRequisição"como um identificador da requisição feita pela aplicação web; e "iResposta"como o identificador da resposta enviada pelo sensor à requisição feita. Todos os identificadores citados são do tipo numérico.

Considerando que foi preciso utilizar quase todo o espaço disponível para o envio de mensagens com metadados, como os identificadores das mensagens recebidas e enviadas (usados para avaliar o desempenho da aplicação, discutida no capítulo 4), foi necessário estabelecer algumas medidas para não ultrapassar a quantidade de 128 *bytes* com os dados do sensor. Assim, convencionou-se que as aplicações que utilizem a API RESTful dos sensores já esperem a unidade na qual o dado será enviado, visando não ser necessário expressar esta informação na mensagem enviada.

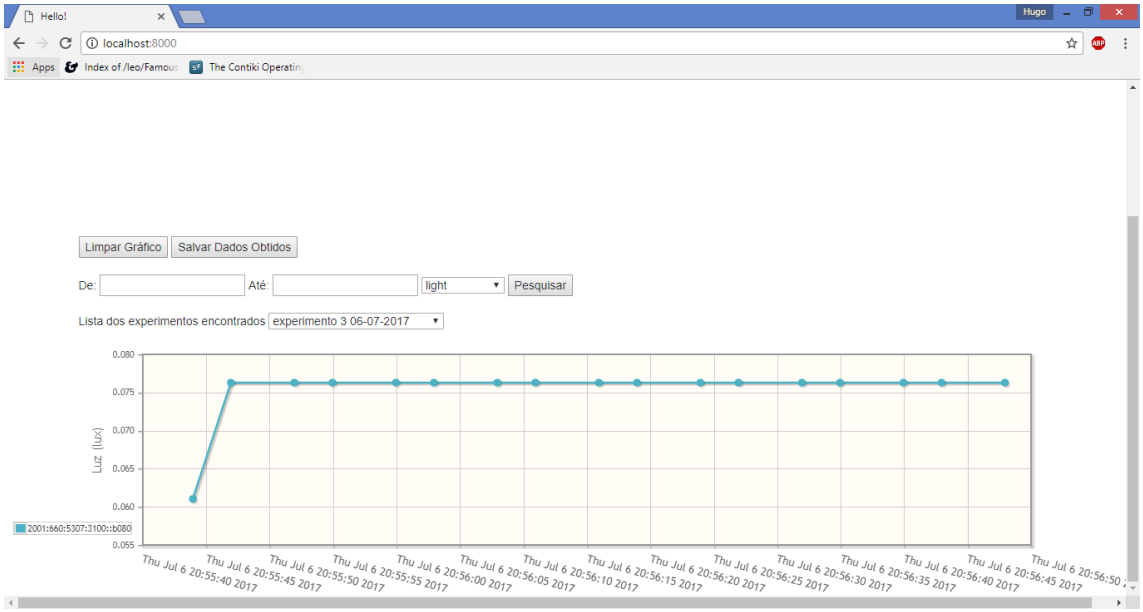

Figura 3.9: Visão de uma observação de um sensor de luminosidade.

Assim, para as medidas de luminosidade, os sensores estão medindo em Lux, e para as medidas de temperatura, em graus Celcius.

3.2.6 Descrição das bibliotecas utilizadas

Para monitorar a leitura dos sensores em tempo real, a interface da aplicação web dispara requisições Ajax para o *backend* da aplicação. Este, por sua vez, é implementado em Python, utilizando Django, um *framework* para aplicações web. Também foi utilizada a biblioteca Coapthon [35], que realiza requisições CoAP (e não HTTP) para o sensor escolhido pelo usuário na aplicação. Neste sentido, a leitura em tempo real constitui-se em requisições Ajax feitas periodicamente, espaçadas de acordo com o tempo determinado pelo usuário da aplicação.

No lado dos sensores, utilizou-se a biblioteca Erbium [22] para implementar um servidor web CoAP. A biblioteca Erbium traz em sua implementação todos os principais recursos do protocolo. O protocolo CoAP emprega o protocolo UDP para camada de transporte, diferentemente de outros protocolos como o HTTP, que emprega o TCP. [21] Observa-se que, uma vez sendo a aplicação voltada à exibição de informações em tempo real, a confiabilidade de entrega garantida por outros

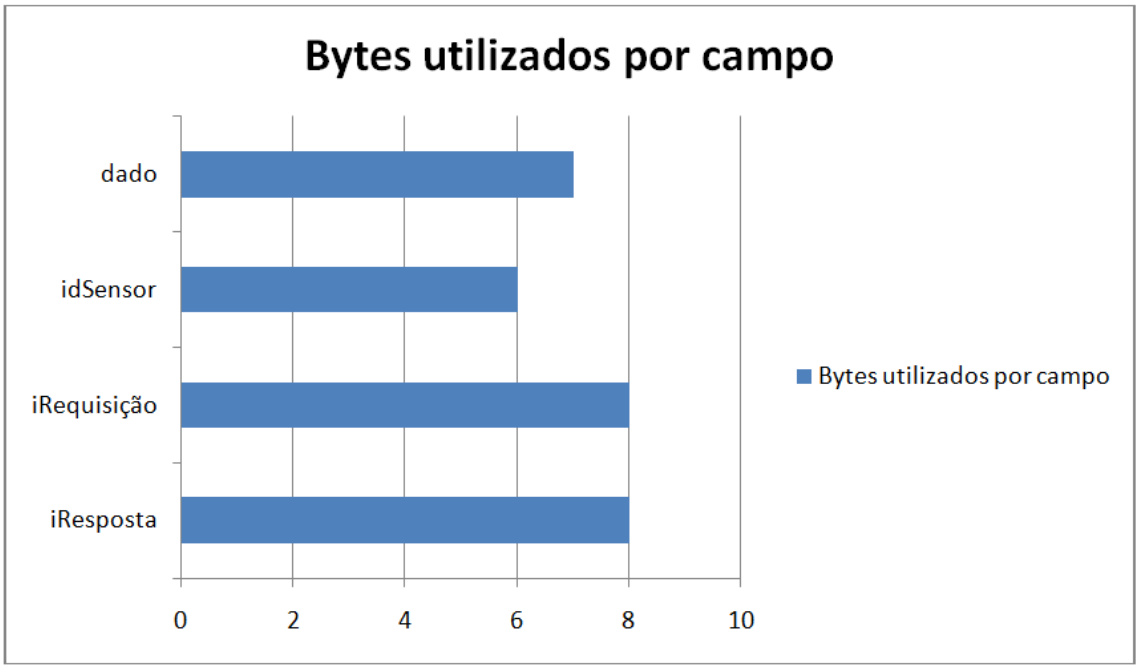

Figura 3.10: Gráfico a respeito da quantidade de bytes necessárias para cada campo da mensagem.

protocolos como o TCP [8] não é tão necessária.

A biblioteca Erbium traz na sua implementação a possibilidade de programar uma API RESTful para o sensor entregando os dados em diversos formatos bem conhecidos como JSON, XML, SOAP e PLAIN TEXT. Neste trabalho, optamos pela utilização de apenas um formato, o formato JSON, a título de simplificação da implementação, e porque a disponibilização de múltiplos formatos aumenta o tamanho de código para além do que os hardwares utilizados podem oferecer em termos de memória não volátil.

Para armazenar as observações obtidas, utilizou-se o banco de dados SQLite[1]. O SQLite implementa um banco de dados embutido, permitindo que aplicações registrem seus dados sem precisar manter um processo SGBD separado. Notável pela sua leveza, simplicidade de utilização, este banco de dados é recomendado para aplicações pequenas, sendo gratuito e de uso livre. O *Framework* Django já traz nativamente ferramentas para lidar com este banco, o que tornou sua escolha bastante vantajosa para este trabalho.

[1] `https://www.sqlite.org/`

Capítulo 4

Avaliação

Nesta seção, apresentamos uma avaliação da solução desenvolvida fazendo uso de um ambiente simulado e de um *testbed* real de RSSFs. Apresentamos, inicialmente, os ambientes de execução utilizados. Em seguida, discutimos a métricas e metodologia adotada para a realização dos experimentos e os resultados obtidos.

4.1 Ambientes de Execução

Dada a natureza distribuída das RSSFs, é fundamental o uso de ambientes controlados (ou parcialmente controlados) para a avaliação das aplicações que executam nessas redes. Apresentamos nesta seção o simulador Cooja e o *testbed* FIT/IoT-LAB usados para avaliação da solução proposta.

4.1.1 Cooja

Cooja [26] é um simulador flexível e extensível, desenvolvido em Java, projetado para simular redes de sensores sem fio. Por simulador flexível, deve-se entender que o Cooja permite trabalhar com as diversas camadas que compõem o sistema, dos dispositivos periféricos das plataformas simuladas ao tráfego de rede. Também é dito que o Cooja é um simulador extensível, pois seu código-fonte é aberto, permitindo que o desenvolvedor desenvolva *plugins* que estendam as funcionalidades do Cooja

para atender a suas próprias necessidades.

Para a montagem das simulações, o Cooja oferece o conceito de "tipo de um nó". Nós de um mesmo tipo executam um mesmo código nos mesmos hardwares periféricos simulados, e têm suas memórias inicializadas da mesma forma. No entanto, mesmo pertencendo a um mesmo tipo, os nós guardam independência de operação, podendo suas memórias diferirem nos dados armazenados conforme a execução da aplicação.

Para realizar uma simulação, o usuário precisa configurar os tipos de nós que deseja utilizar, e isto envolve a escolha por um hardware da lista de hardwares compatíveis com o Contiki. Para o *firmware*, o usuário pode escolher o código fonte e utilizar a interface do Cooja para a compilação do mesmo, podendo ainda contar com as mensagens do compilador, como os avisos (*warnings*) e mensagens de erro.

O usuário possui, então, total liberdade para montar a sua simulação, escolhendo quantos tipos de nós quanto deseje e a quantidade de nós para cada tipo. Para isto, o Cooja oferece uma interface gráfica para ilustrar o mapa da rede de sensores. É através desta que o usuário poderá visualizar diversos aspectos da rede e dos sensores, como os leds de cada um dos sensores e a distância entre um sensor e os sensores vizinhos. Enquanto a simulação é executada, o tráfego na rede é representado através de setas, ilustrando o remetente e o destinatário de cada mensagem enviada. A Figura 4.1 ilustra a interface de simulação do Cooja.

4.1.2 Testbed FIT/IoT-LAB

Existe uma diversidade de *testbeds* com foco em RSSFs e IoT. Para bem comportar simulações de RSSFs e suas diversas aplicações, é esperado que essas plataformas sejam suficientemente flexíveis para permitir variedade de hardwares e topologias de rede, e garantir a repetição dos experimentos [37].

Este trabalho utilizou o *testbed* FIT IoT-LAB, desenvolvido pelo FIT (*Future of Internet of Things*), um consórcio de universidades europeias que se dedica ao

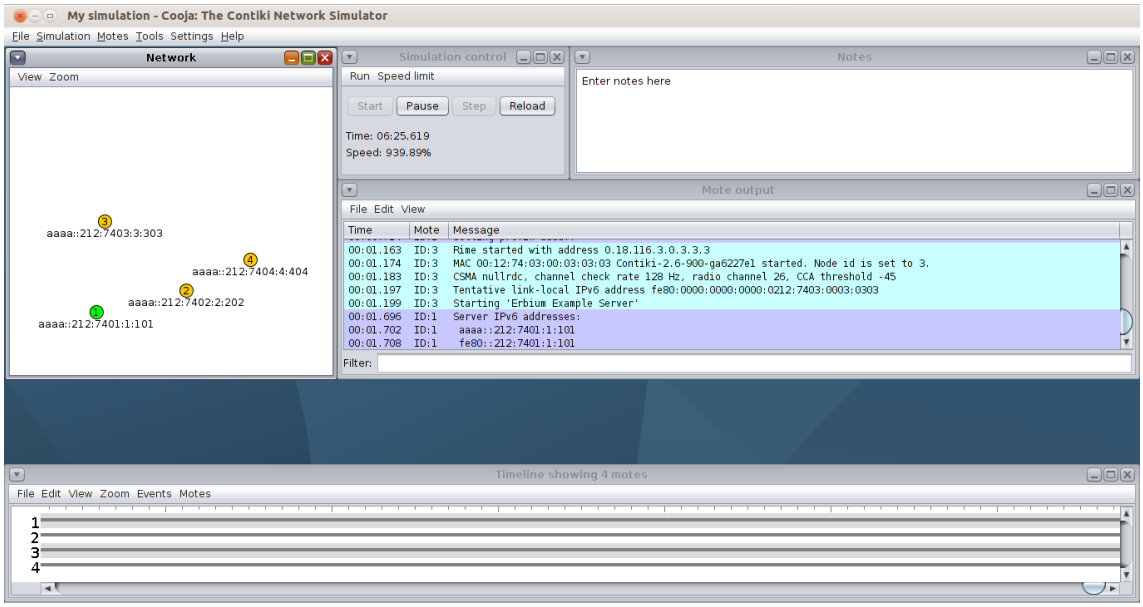

Figura 4.1: Simulação em andamento de um experimento no Cooja.

desenvolvimento de *testbeds* para a comunidade científica [1]. Este *testbed* está localizado na França, apresentando vários sítios em várias localidades deste país, como Lion e Grenoble.

São características do IoT-LAB, enquanto plataforma experimental de RSSF e IoT, um ambiente rico e heterogêneo e a capacidade de gerência, interação e monitoramento dos experimentos. [2]

A plataforma oferece uma variedade de sensores sem fio, dotados de diferentes arquiteturas e processadores. Para a utilização do sistema, o usuário poderá dispor do portal web ou da API REST oferecida através de linha de comando e de acesso através de SSH[2].

Independente de como o usuário acesse o *testbed*, para rodar um experimento é preciso que o usuário reserve um ou mais nós e uma duração para a sua reserva. Neste momento é possível fazer as configurações necessárias para experimento que se deseja executar, como a fonte de alimentação dos nós, o *firmware* que cada nó deve executar, assim como a escolha de ferramentas adicionais de monitoramento das métricas de avaliação, como o consumo de energia em cada um dos nós. O

[1] `https://www.iot-lab.info/about-us/`

[2] `https://www.ietf.org/rfc/rfc4251.txt`

experimento, então, será inicializado o mais cedo possível, ou em uma determinada data, conforme especificação do usuário.

4.2 Plataformas utilizadas

Da lista de plataformas compatíveis com o Contiki no ambiente Cooja, este trabalho utilizou a plataforma Tmote Sky. São especificações do Tmote Sky [6]:

- microcontrolador MSP430;
- 48kB de memória interna Flash;
- 1 MB de memória externa Flash;
- sensores de umidade, temperatura e luminosidade.

Já no Testbed IOT-LAB, utilizamos a plataforma M3 [3]. São especificações da plataforma M3:

- microcontrolador ARM Cortex M3;
- 64kB de RAM;
- 128 Mb de memória externa Flash;
- sensores de luminosidade, pressão atmosférica, temperatura e acelerômetro.

O *testbed* oferecia, além da plataforma M3, outras dois hardwares, o WSN430, dispositivo de muito baixo poder computacional baseado na plataforma MSP430, e o A8 [4], que utiliza o mesmo ARM Cortex da plataforma M3, porém diferindo nos sensores disponíveis e no fato de executar Linux em vez de Contiki.

A escolha por essas plataformas se deu em função de serem as que apresentavam a maior quantidade de sensores úteis para a aplicação web final. No caso do *testbed*,

[3] `https://www.iot-lab.info/hardware/m3/`
[4] `https://www.iot-lab.info/hardware/a8/`

não era possível empregar a plataforma WSN430, embora ela apresentasse o hardware ideal para este trabalho, pois este não foi incluído nas sub-redes disponíveis para o acesso IPv6. Já a plataforma A8 não apresentava os sensores de condições ambientes requeridos, motivo pelo qual no *testbed* optou-se por usar a plataforma M3.

4.3 Métricas usadas e metodologia

Os testes realizados nos ambientes de execução descritos na sessão anterior buscaram de uma forma geral avaliar a aplicação através dos dados gerados pela mesma. Partindo desse princípio, buscou-se avaliar a influência do intervalo entre as requisições sobre a confiabilidade das informações obtidas, em particular em relação à perda de mensagens. Isso justifica-se pois a aplicação é pensada para um contexto de tempo real de resposta, então é adequado quantificar o quão próximo do real são as informações apresentadas para o usuário. Variando o tempo entre as requisições de 1 a 0,1 segundo, contabilizou-se quantas mensagens chegaram durante um período de 3 minutos. Para cada intervalo entre as requisições, repetiu-se o experimento por 10 vezes, sendo os resultados apresentados a média dos desses valores. Esta seção mostra os resultados obtidos a partir dessas experimentações.

4.3.1 Avaliação da quantidade requisições feitas e recebidas

Aproveitando o fato de que a API dos sensores emprega o formato JSON como padrão nas respostas das requisições feitas pelo usuário através da aplicação web, foi possível transmitir mais facilmente as informações necessárias para identificar que requisição e que resposta foram processados pelo sensor.

Para isto, tanto a aplicação quanto cada um dos sensores foram dotados de contadores internos iterativos. A cada requisição que a aplicação disparava a um dos sensores, um número inteiro identificando esta requisição era enviado através da própria *query string* da URL. O sensor, ao receber a requisição, extrai o valor da variável da *query string* e o insere no JSON de resposta, assim como também insere

o valor do seu contador interno, um número inteiro que identifica esta resposta. Assim, ao chegar novamente no cliente, a mensagem traz a informação completa dos identificadores da requisição e da resposta. Ao visualizar-se a lista de todas as mensagens recebidas, descontinuidades na numeração das mensagens acusaria a perda de informação durante o tráfego da mensagem na rede.

4.4 Resultados e avaliação

Esta seção avalia o desempenho da solução proposta, com base nos resultados obtidos. Através da observação do comportamento do sistema nos dois ambientes utilizados, será discutido como esses resultados podem limitar ou influenciar a utilização da aplicação web pelo usuário final.

4.4.1 Ambiente Cooja

Foram realizados dois experimentos com o simulador Cooja: um mais simples, contendo apenas dois sensores, sendo um com a função de roteador de borda e o outro rodando a aplicação; e outro mais elaborado, contendo 9 sensores cobrindo uma área de aproximadamente $100m^2$.

Os experimentos envolvendo dois sensores apresentaram boa porcentagem de entrega de mensagens, se mantendo em média acima de 75% para períodos de aquisição a partir de 0,6 segundo. O experimento mostrou que o limite mínimo de intervalo de aquisição é de 0,4 segundo, já que a aplicação deixa de responder ao escolher-se intervalos menores do que este. Abaixo de 0,7 segundo, observa-se um aumento do número de mensagens perdidas. Observou-se uma média de 1 requisição e 1 resposta perdida para cada mensagem recebida com sucesso. A Tabela 4.1 mostra o período entre as aquisições, o número de mensagens esperadas, a média do número de mensagens recebidas acrescida do intervalo de confiança e o desvio padrão da amostra. Todos os cálculos relacionados ao intervalo de confiança foram feitos com base em um nível de confiança de 95%.

Os experimentos envolvendo 9 sensores registraram uma queda considerável de

Período (s)	Num. msgs esperadas	Msgs recebidas	Desvio padrão
1	180	$173,4 \pm 4,55$	7,35
0,9	200	$196,9 \pm 0,51$	0,83
0,8	225	$213,9 \pm 7,82$	12,63
0,7	257	$239,7 \pm 1,88$	3,03
0,6	300	$229,6 \pm 2,06$	3,32
0,5	360	$247 \pm 20,60$	33,24
0,4	450	$262,9 \pm 49,86$	80,45

Tabela 4.1: Avaliação de mensagens no ambiente Cooja para dois sensores.

desempenho. A aplicação só conseguiu operar com períodos de até 0,6 segundo, sendo constantes travamento tanto na aplicação quanto na simulação. Todos os testes foram executados requisitando-se dados do sensor identificado na Figura 4.2 com o número 3. A Tabela 4.2 mostra os valores obtidos.

Período (s)	Num. msgs esperadas	Num. msgs recebidas	Desvio padrão	Req. Perdidas	Resp. Perdidas
1	180	$93,7 \pm 12,59$	20,30	4,2	33,3
0.9	200	$90,7 \pm 13,16$	21,23	3,8	38
0.8	220	$114,4 \pm 12,01$	19,38	3,6	39,7
0.7	257	$125,3 \pm 17,05$	27,52	4,2	44,7
0.6	300	$105,7 \pm 15,26$	24,62	9,6	37,6

Tabela 4.2: Avaliação de mensagens no ambiente Cooja para 9 sensores.

4.4.2 Ambiente FIT/IOT-LAB

Os experimentos realizados no *testbed* envolveram um total de 6 sensores, sendo um deles usado como roteador de borda e o restante para o sensoriamento do ambiente. Neste experimento conseguiu-se o limite mínimo de 0,3 segundo, um desempenho médio superior ao que foi obtido nas simulações, possivelmente explicado pelo fato de ter sido utilizado uma placa de maior desempenho computacional.

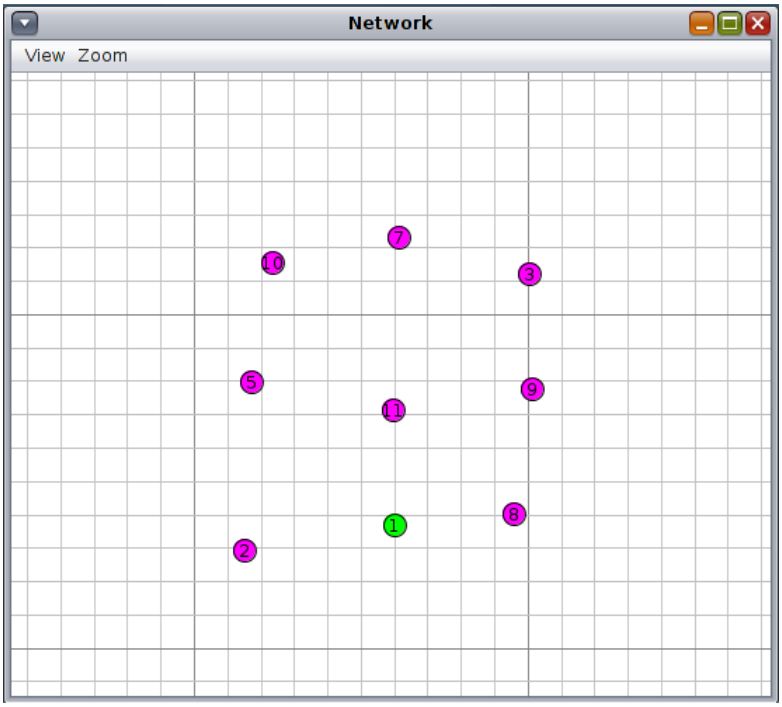

Figura 4.2: Topologia dos testes envolvendo 9 sensores, sendo o roteador de borda identificado pelo número 1.

Período (s)	Num. msgs esperadas	Num. msgs recebidas	Desvio padrão	Req. Perdidas	Resp. Perdidas
1	180	$179,2 \pm 1,10$	1,77	0,8	1,5
0.9	200	$195,5 \pm 1,44$	2,33	3,1	31
0.8	220	$205,8 \pm 15,88$	25,62	8	8,6
0.7	257	$237,5 \pm 5,70$	9,20	16,4	17,1
0.6	300	$256,7 \pm 14,41$	23,26	40,6	41,1
0.5	360	$252,9 \pm 40,37$	65,14	102,7	104,1
0.4	450	$196,7 \pm 12,31$	19,86	244,6	245
0.3	600	$218,4 \pm 10,63$	17,15	374,1	374

Tabela 4.3: Avaliação de mensagens no ambiente IOT-LAB.

Capítulo 5

Conclusão

Este trabalho buscou o desenvolvimento de uma solução automatizada para o sensoriamento remoto das condições ambientais dos postos de trabalho, buscando a sua adequação às condições descritas pelas Normas Regulamentadoras aprovadas pelo Ministério do Trabalho.

Sabe-se que a Norma recomenda que as condições referidas sejam medidas em cada posto de trabalho, por técnicos especializados, para compor relatórios que atestem que as empresas seguem o que as leis trabalhistas prescrevem. Contudo, esta tarefa pode ser longa e exaustiva, e a presença do técnico e seu deslocamento por todo o ambiente pode tornar essa tarefa lenta.

Assim, pensou-se em uma alternativa para realizar essas medidas remotamente, permitindo a visualização das condições de trabalho em tempo real, não sendo necessária a presença física das pessoas para tal.

Para o monitoramento, buscou-se utilizar uma RSSF, posicionando-se os sensores em cada um dos ambientes que se deseja monitorar. Essa RSSF deve estar integrada à Internet, sendo cada um dos sensores acessíveis através de requisições externas por meio de endereços IPv6. Esta infraestrutura é vantajosa, pois além de empregar um protocolo amplamente utilizado na Internet, tornando mais natural a integração das duas redes, ela não está vinculada a apenas um tipo de utilização, sendo independente e podendo ser utilizada por diferentes aplicações que queiram consultar os

dados que ela oferece, neste caso, as medidas de temperatura, luminosidade, ruído, velocidade do ar e umidade relativa.

Para a visualização dos dados obtidos pelos sensores, desenvolveu-se uma aplicação web, cujas principais funcionalidades são a visualização de um gráfico mostrando em tempo real a variação do valor da medida requisitada no tempo, bem como a possibilidade de registrar essa observação e pesquisar observações feitas anteriormente.

Para a implementação da solução, foram utilizados bibliotecas consagradas pelo amplo uso por parte da comunidade desenvolvedora. Na RSSF, utilizou-se o sistema operacional Contiki, desenvolvido propriamente para ser executado em hardwares de baixo poder computacional. Cada sensor executa uma aplicação simples que disponibiliza as informações das medições através de API RESTful, empregando para isto o protocolo CoAP, uma versão protocolo HTTP para RSSF e IoT, mas que implementa outros recursos apropriados para o seu contexto. Já a aplicação web foi desenvolvida utilizando o *framework* Django, o banco de dados SQLite, assim como a biblioteca CoAPthon, para realizar requisições CoAP.

Para a realização de testes na solução desenvolvida, foram utilizados dois ambientes distintos: o simulador Cooja e o *testbed* FIT/IoT-LAB, que oferece uma experimentação mais próxima do uso final, pois permite o acesso à sensores físicos funcionando em tempo real.

Os experimentos mostraram que os dados podem ser visualizados a contento, estando o limite mínimo de tempo entre as requisições bem menor do que aquele que na prática os usuários poderiam solicitar, sendo este na ordem de décimos de segundo. Algumas perdas foram observadas nos dois ambientes de experimentação. Porém, o percentual apresentado não invalida a solução desenvolvida.

Este trabalho foi instrutivo, pois revelou alguns aspectos interessantes sobre o atual estágio de desenvolvimento de soluções que integram aplicações web e aplicações executadas por RSSFs. O desenvolvedor de uma aplicação precisa lidar com uma diversidade de dispositivos, *drivers* e bibliotecas, o que gasta tempo que pode-

ria estar sendo investido no desenvolvimento da aplicação em si. Assim, os sistemas operacionais dedicados a dispositivos inteligentes poderiam oferecer ao desenvolvedor interfaces de programação mais amigáveis, permitindo uma maior abstração dos detalhes de implementação de *drivers* e outros códigos de baixo nível. O desenvolvedor, assim, concentraria mais seus esforços no desenvolvimento das regras de negócio da sua solução, quadro este que certamente estimularia ainda mais o uso das tecnologias envolvendo RSSFs e IoT.

Essas considerações também são válidas para os ambientes de experimentação focados em Internet das Coisas. O *testbed* utilizado neste trabalho, o IOT-LAB, possui uma série de dificuldades para a experimentação de uma rede de sensores heterogênea, isto é, na utilização de diferentes dispositivos em um mesmo experimento, já que nem todas as placas que este *testbed* oferece estão conectadas em rede. Isto diminuiu bastante as possibilidades de experimentação do sistema desenvolvido neste trabalho, já que então só foi possível trabalhar com apenas um modelo de placa.

Um ponto importante a ser destacado é a documentação escassa das bibliotecas desenvolvidas para Contiki, assim como do próprio Contiki. Isto torna muito mais árduo, por exemplo, o entendimento dos diversos códigos extras que o IOT-LAB emprega para a utilização do Sistema Operacional em seus experimentos, o que acaba por praticamente limitar a sua utilização ao que é mostrado nos numerosos códigos de exemplo que acompanham o sistema.

Assim, a maior parte do tempo que este trabalho demandou foi investida no entendimento dos códigos de exemplo encontrados tanto junto com o código fonte do Contiki, quanto nos tutoriais no portal do FIT/IoT-LAB, e não no desenvolvimento propriamente dito. Em termos práticos, a única fonte de informação segura para o desenvolvedor ainda é a leitura direta destes códigos, já que também a comunidade não é tão ampla nem sempre acessível.

Durante este trabalho, diversas ideias tiveram de ser adaptadas, pois ao compilar-se os códigos desenvolvidos, não raro o tamanho do *firmware* ultrapassada a memória disponível nas plataformas utilizadas, tanto no simulador quanto no testbed. Os có-

digos fonte que demonstram a implementação de *web services* utilizando HTTP e CoAP trazem implementados diversos recursos, mas cada um desses recursos não necessitam de muita memória nem de muito processamento, pois geralmente envolvem o retorno de uma *string* fixa ou de um número gerado aleatoriamente. Ao utilizar-se os *drivers* reais para realizar a leitura de luz e temperatura, foi necessário ser o mais econômico possível no restante do código. O desenvolvimento de aplicações para RSSFs e IoT precisa necessariamente considerar os dispositivos alvo do sistema.

5.1 Trabalhos Futuros

Sobre as aplicações desenvolvidas no decorrer deste trabalho, tanto a que é executada diretamente nos dispositivos sensores quanto a aplicação WEB, consideramos que a solução desenvolvida é estável e completa, porém é possível evoluí-la ainda mais.

Trabalhos futuros podem incluir, em relação à regras de negócio, o oferecimento do monitoramento de todas as grandezas necessárias para o cumprimento total do que prescreve a NR 17 em relação a ambientes de trabalho, isto é, o monitoramento total das condições de luminosidade, temperatura, umidade do ar e pressão sonora, além da possibilidade de se produzir relatórios oficiais baseados nos dados obtidos. Este seria um desafio, tanto no sentido de conjugar todos os sensores necessários quanto avaliar se os sensores cumprem os padrões aceitos pelas certificações competentes para a emissão de relatórios de valor oficial. Essa demanda também traz a necessidade de evolução da API RESTful oferecida pela aplicação executada nos sensores, para que ofereça dados em outros formatos além do JSON, podendo também conter outras informações importantes, como a unidade das medidas realizadas, admitindo-se que as medidas possam ser realizadas em diferentes unidades. Essa evolução, contudo, precisa ser feita de maneira estratégica, para não onerar a capacidade operacional dos sensores, já muito comprometida com a execução do servidor CoAP.

Em relação às tecnologias utilizadas, seria interessante experimentar a adoção de

web sockets, tecnologia hoje largamente empregada em sistemas colaborativos e cujas informações são disponibilizadas em tempo real. A implementação de *web sockets*, tanto na aplicação web quanto nos sensores, respeitando a limitação computacional destes, traria benefícios como a redução no tráfego da rede, permitindo maior fluidez nos dados exibidos.

Referências

[1] Aarthi, R., e Renold, A. P. CoAP based acute parking lot monitoring system using sensor networks. *ICTACT journal on communication technology: special issue on advances in wireless sensor networks 5*, 02 (2014).

[2] Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., e others. FIT IoT-LAB: A large scale open experimental IoT testbed. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on* (Milão, Itália, 2015), IEEE, pp. 459–464.

[3] Alves, P. M. Aplicação da NR 17 em uma enfermaria de uma santa casa de misericórdia: um estudo de caso. Tese de Mestrado, Universidade Estadual Paulista, São Paulo, 2010.

[4] Antunes, V. B. Roteamento sensível ao contexto em redes de sensores sem fio: Uma abordagem baseada em regras de aplicação para o protocolo RPL. Tese de Mestrado, Universidade Federal do Espírito Santo, Vitória, 2014.

[5] Belcredi, G., Modernell, P., Sosa, N., Steinfeld, L., e Silveira, F. An implementation of a home energy management platform for smart grid. In *Innovative Smart Grid Technologies Latin America (ISGT LATAM), 2015 IEEE PES* (Montevidéu, Uruguai, 2015), IEEE, pp. 270–274.

[6] Boratti, R. Internet das coisas: estudo do simulador COOJA. Tese de Mestrado, Universidade Federal De Santa Catarina, Santa Catarina, 2014.

[7] Cabral, F. M. Análise da demanda ergonômica, medição de iluminância e temperatura em um supermercado, 2013. Monografia de Especialização.

[8] Cerf, V., Dalal, Y., e Sunshine, C. Specification of internet transmission control program. RFC 675, RFC Editor, December 1974. `http://www.rfc-editor.org/rfc/rfc675.txt`.

[9] Colitti, W., Steenhaut, K., e Caro, N. D. Integrating wireless sensor networks with the web. In *In Proceedings of Workshop on Extending the Internet to Low power and Lossy Networks* (Chicago, Illinois, 2011).

[10] DANTAS, H. M. SiMoni: um Sistema de Monitoramento de ambientes de trabalho. `https://github.com/hmdantas/simoni`.

[11] Deering, S. E., e Hinden, R. M. Internet protocol, version 6 (IPv6) specification. RFC 2460, RFC Editor, December 1998. `http://www.rfc-editor.org/rfc/rfc2460.txt`.

[12] Delicato, F. C. *Middleware baseado em serviços para redes de sensores sem fio*. PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

[13] Dunkels, A., Grönvall, B., e Voigt, T. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)* (Tampa, Flórida, Estados Unidos, novembro de 2004).

[14] Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., e Berners-Lee, T. Hypertext transfer protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999. `http://www.rfc-editor.org/rfc/rfc2616.txt`.

[15] Fielding, R. T., e Taylor, R. N. Architectural styles and the design of network-based software architectures. Tese de Mestrado, University of California, Irvine, California, Estados Unidos, 2000.

[16] Gubbi, J., Buyya, R., Marusic, S., e Palaniswami, M. Internet of things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems 29*, 7 (2013), 1645–1660.

[17] Guinard, D., Trifa, V. M., e Wilde, E. Architecting a mashable open world wide web of things, 2010. Relatório técnico.

[18] Ishaq, I., Carels, D., Teklemariam, G. K., Hoebeke, J., Van den Abeele, F., De Poorter, E., Moerman, I., e Demeester, P. IETF standardization in the field of the internet of things (IoT): a survey. *Journal Of Sensor And Networks*, 2 (2013), 235–287.

[19] Jeferson Rodrigues Cotrim, J. H. K. Avaliação de desempenho do protocolo RPL em ambientes com mobilidade. *Simpósio Brasileiro de Telecomunicações* (2016).

[20] Kovatsch, M. Demo abstract: Human–coap interaction with copper. In *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2011)* (Barcelona, Spain, junho de 2011).

[21] Kovatsch, M., Duquennoy, S., e Dunkels, A. A low-power CoAP for contiki. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on* (Valência, Espanha, 2011), IEEE, pp. 855–860.

[22] Kovatsch, M., Duquennoy, S., e Dunkels, A. A low-power coap for contiki. In *Proceedings of the 8th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2011)* (Valência, Espanha, outubro de 2011).

[23] Kushalnagar, N., Montenegro, G., e Schumacher, C. IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals. RFC 4919, RFC Editor, August 2007. `http://www.rfc-editor.org/rfc/rfc4919.txt`.

[24] Loureiro, A. A., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A. d. F., Nakamura, E. F., e Figueiredo, C. M. S. Redes de sensores sem fio.

In *Simpósio Brasileiro de Redes de Computadores (SBRC)* (Natal, 2003), sn, pp. 179–226.

[25] Ma, X., e Luo, W. The analysis of 6lowpan technology. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on* (Wuhan, China, 2008), vol. 1, IEEE, pp. 963–966.

[26] Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., e Voigt, T. Cross-level sensor network simulation with Cooja. In *Local computer networks, proceedings 2006 31st IEEE conference on* (Tampa, Flórida, Estados Unidos, 2006), IEEE, pp. 641–648.

[27] Pandesswaran, C., Surender, S., e Karthik, K. Remote patient monitoring system based CoAP in wireless sensor networks. *Int J Sens Netw Data Commun 5*, 145 (2016), 2.

[28] Paulson, L. D. Building rich web applications with Ajax. *Computer 38*, 10 (2005), 14–17.

[29] Postel, J. User datagram protocol. STD 6, RFC Editor, August 1980. `http://www.rfc-editor.org/rfc/rfc768.txt`.

[30] Romkey, J. Nonstandard for transmission of IP datagrams over serial lines: SLIP. STD 47, RFC Editor, June 1988.

[31] Shelby, Z., e Bormann, C. *6LoWPAN: The wireless embedded Internet*, vol. 43. John Wiley & Sons, Grã-Bretanha, 2011.

[32] Shelby, Z., Hartke, K., e Bormann, C. The constrained application protocol (CoAP). RFC 7252, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7252.txt`.

[33] Simpson, W. The point-to-point protocol (PPP) for the transmission of multi-protocol datagrams over point-to-point links. RFC 1331, RFC Editor, May 1992.

[34] Sousa, N. d. C., Ricardo de Andrade, L., Araújo, H. d. S., Lima, J. C. C., e Filho, R. H. Uma otimização do protocolo RPL para aplicaç oes de internet das coisas. *Anais da III Escola Reginal de Informática do Piauí* (2017).

[35] Tanganelli, G., Vallati, C., e Mingozzi, E. Coapthon: Easy development of CoAP-based IoT applications with python. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on* (Milão, Itália, 2015), IEEE, pp. 63–68.

[36] Taube, A. P., e Barja, P. R. Estudo acústico de ambientes hospitalares: Unidade de terapia intensiva (uti). In *XII Congresso Latino-Americano de Iniciação Científica e VIII Encontro de Pós-Graduação-Energia: geração, uso e consequência* (São José dos Campos, 2008).

[37] Tonneau, A.-S., Mitton, N., e Vandaele, J. A survey on (mobile) wireless sensor network experimentation testbeds. In *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on* (Marina Del Rey, California, 2014), IEEE, pp. 263–268.

[38] Tsvetkov, T. RPL: IPv6 routing protocol for low power and lossy networks. *Network Architectures and Services 59* (2011).

[39] Wachowicz, M. C. *Ergonomia.* Guia de Estudos para Ensino na modalidade em EaD. Ministério da Educação, Rio de Janeiro, 2013.

[40] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., e Alexander, R. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550, RFC Editor, March 2012. `http://www.rfc-editor.org/rfc/rfc6550.txt`.

[41] Yick, J., Mukherjee, B., e Ghosal, D. Wireless sensor network survey. *Computer networks 52*, 12 (2008), 2292–2330.