UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
TECNÓLOGO EM REDES DE COMPUTADORES

José Carlos da Silva de Carvalho

INOSHARE: GERENCIAMENTO DE APLICAÇÕES EM REDES DE SENSORES E ATUADORES COMPARTILHADOS

QUIXADÁ
2014

José Carlos da Silva de Carvalho

INOSHARE: GERENCIAMENTO DE APLICAÇÕES EM REDES DE SENSORES E ATUADORES COMPARTILHADOS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Tecnólogo em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: computação

Orientadora Prof[a]. Atslands Rego da Rocha
Coorientador: Prof. Dr. Flávio Rubens de Carvalho Sousa

QUIXADÁ
2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

C324i Carvalho, José Carlos da Silva de
INOSHARE: gerenciamento de aplicações em redes de sensores e atuadores compartilhados / José Carlos da Silva de Carvalho.– 2014.
70 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Tecnologia em Redes de Computadores, Quixadá, 2014.
Orientação: Prof. Dra. Atslands Rego da Rocha
Área de concentração: Computação

1. Computação em Nuvem 2. Redes de Sensores Sem Fio 3. Interfaces de usuário (Sistema de computador) I. Título.

CDD 004.36

José Carlos da Silva de Carvalho

INOSHARE: GERENCIAMENTO DE APLICAÇÕES EM REDES DE SENSORES E ATUADORES COMPARTILHADOS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Tecnólogo em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: computação

Aprovado em: _____ / novembro / 2014.

BANCA EXAMINADORA

_____
Profª. Dª. Atslands Rego da Rocha (Orientadora)
Universidade Federal do Ceará-UFC

_____
Prof. M.e. Marcos Dantas Ortiz
Universidade Federal do Ceará-UFC

_____
Prof. Dr. Flávio Rubens de Carvalho Sousa
Universidade Federal do Ceará-UFC

A minha doce esposa...

AGRADECIMENTOS

À minha doce esposa Nathyelle Melo, por sempre me lembrar que eu sou capaz de tudo se eu tiver força de vontade e amor, e que sempre esteve do meu lado acreditando em minha capacidade de alcançar quaisquer objetivos.

Ao meu amado filho Levih que por muitas vezes esteve no meu colo acompanhando a digitação desse trabalho, e que me trouxe em muitos momentos sorrisos e abraços que me alegravam em momentos difíceis.

À Profª Drª Atslands Rego da Rocha por acreditar em minha capacidade, ter me orientado e guiado diante de inúmeros desafios encontrados no caminho, sendo para mim uma referência em profissionalismo e comprometimento.

Aos pais de minha esposa, Franscisco ("tchesco") e Joyce por serem como meus segundos pais, me dando apoio, força em momentos difíceis, sendo pessoas nas quais eu me orgulho de fazerem parte da minha família.

Aos professores da banca de defesa, Prof. MSc. Marcos Dantas e Prof. Dr. Flávio Rubens, por terem acompanhado a evolução desse trabalho e pelas valiosíssimas dicas que foram fundamentais para a realização do trabalho.

Aos colegas da universidade por compartilharem as alegrias e dificuldades desses anos de convívio, além de ajudar na construção do conhecimento que resultaram nessas páginas.

"Sonhe alto. O máximo que pode acontecer
é você realizar um sonho à altura."
(Pedro Gabriel)

RESUMO

Nos últimos anos, as redes de sensores evoluíram de simples aplicações dedicadas para infraestruturas integradas compartilhadas por várias aplicações, característica que constitui as redes de sensores e atuadores compartilhados. Com o advento da computação em nuvem, são oferecidos recursos computacionais por demanda de forma escalável, com alta disponibilidade e com custo reduzido. Com o intuito de proporcionar um melhor aproveitamento dos recursos existentes em uma rede de sensores e atuadores através do uso da computação em nuvem, esse trabalho apresenta o INOSHARE. A execução das aplicações criadas no INOSHARE é realizada na nuvem, fornecendo um melhor desempenho de execução, armazenamento dos dados coletados e disponibilidade.Com o INOSHARE, os usuários podem criar e supervisionar suas aplicações via interface web, facilitando assim a gerência das aplicações criadas. Tendo em vista que os dados coletados por redes de sensores podem, a médio prazo, alcançar um grande volume, o INOSHARE armazena os dados coletados pela rede de sensores em bancos de dados não relacionais, que oferecem maior desempenho em comparação com bases de dados relacionais na existência de uma grande quantidade de dados. Os dados coletados ainda são disponibilizados para o uso fora do INOSHARE, podendo ser reaproveitados para aplicações *mobile* e aplicações que utilizem dados históricos.

Palavras chave: rede de sensores compartilhadas, computação em nuvem, arduino

ABSTRACT

In recent years, sensor networks are evolving from simple applications to integrated infrastructure that can be shared by several applications, a key characteristic of the shared sensors and actuators networks. With the advent of cloud computing, computing resources are offered by demand for scalable and highly available. Cloud computing allows us to consume computing resources on demand and cost. In order to provide the best use of existing resources on a network of sensors and actuators, this work presents the INOSHARE. The execution of applications created in INOSHARE is performed in the cloud, providing better performance for the application, the collected data storage and availability. With INOSHARE, users can create and supervise their applications via web interface, thus facilitating the management of the created applications. Given that the data collected by sensor networks can in the medium term, reach a large volume, the INOSHARE stores the data collected by the sensor network in non-relational databases, which offer higher performance compared to relational databases the existence of a large amount of data. The collected data is still available for use outside the INOSHARE and can be reused for mobile applications and applications using historical data.

Keywords: shared sensor network, cloud computing, arduino.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente um número crescente de redes de sensores sem fio (RSSFs) estão sendo utilizadas para coletar informações sob uma ampla gama de aplicações. Dados coletados por redes de sensores são cruciais para monitorar condições ambientais e também é uma tecnologia que pode ser utilizada no desenvolvimento de aplicações sensíveis ao contexto. Uma RSSF pode ser formada por um conjunto de dispositivos que possuem recursos limitados de energia e armazenamento (ROCHA, 2012). As RSSF que possuem atuadores, se diferenciam das RSSF tradicionais pelo fato de que, alguns nós podem conter dispositivos capazes de executar tarefas de automação, como ligar e desligar equipamentos eletrônicos. Tais redes são conhecidas como redes de sensores e atuadores sem fio (RSASF). Nos últimos anos, as RSASF evoluíram de plataformas específicas dedicadas para infraestruturas integradas compartilhadas por várias aplicações. Redes de sensores e atuadores compartilhados (RSACs) oferecem grandes vantagens em termo de flexibilidade e custo, uma vez que permitem o compartilhamento de recursos e a alocação dinâmica entre múltiplas aplicações (BHATTACHARYA, 2010).

Tendo em vista que a massa de dados coletados por uma RSAC pode crescer rapidamente e esses dados podem ser acessados por diversas aplicações, o uso de um banco de dados relacional, que armazenam os dados em tabelas, pode a médio prazo apresentar uma significativa queda de desempenho nas aplicações (DIALLO, 2012). Bancos de dados NoSQL provêem alta escalabilidade e desempenho elevado em comparação a base de dados relacionais (WANG & TANG, 2012).

Com o avanço da internet e o surgimento da computação em nuvem, apareceram diversos tipos de aplicações resultando em uma maior demanda para os bancos de dados. Requisitos como a alta escalabilidade, máximo desempenho e alta disponibilidade são cada vez mais necessários em bancos de dados. Apesar dos bancos de dados relacionais possuírem uma alta capacidade de armazenamento, com o passar do tempo têm enfrentado problemas como a dificuldade de expansão, apresentando um desempenho inferior frente a atual demanda (G. Le *et all,* 2011). Uma grande diversidade de tipos de bancos de dados surgiram para enfrentar os diversos desafios existentes na atual demanda de dados. Em geral, os novos modelos de bases de dados são muito diferentes de bases de dados relacionais tradicionais, por isso é mencionado como banco de dados "NoSQL". NoSQL também pode ser interpretado como a abreviatura de "NÃO SÓ SQL" (G. Le *et all,* 2011).

O gerenciamento de dados de redes de sensores encara desafios como: escalabilidade, disponibilidade dos dados coletados e flexibilidade (DIALLO, 2012). Tais problemas são herdados pelas RSACs já que são um tipo de redes de sensores e os dados coletados ainda podem ser gerenciados por múltiplos usuários. Os desafios existentes podem ser amenizados com o uso da computação em Nuvem. A computação em Nuvem se caracteriza por ser uma tecnologia capaz de fornecer flexibilidade, escalabilidade em serviços de forma virtualizada, armazenamento em massa de forma eficiente e de simples gerenciamento, com custo baseado no uso (ALAMRI, 2013). Muitos trabalhos já foram realizados e oferecem soluções para integração de redes de sensores com a Computação em Nuvem. O foco desse tipo de aplicação é fundamentado no processamento dos dados a fim de suprir as necessidades computacionais existentes nos sensores (AHMED, 2011; ALAMRI, 2013; SHAH, 2013).

Nesse contexto, dados os desafios encontrados no processamento e gerenciamento de RSACs, a utilização da Computação em Nuvem no armazenamento e execução das aplicações, pode ajudar a suprir as necessidades de processamento, armazenamento dos dados coletados e flexibilidade. Nesse trabalho apresentamos uma solução para o melhor aproveitamento dos recursos de uma RSAC, utilizando a computação em nuvem para obter disponibilidade dos dados e efetuar o processamento das aplicações.

2 TRABALHOS RELACIONADOS

A abordagem de RSAC foi inicialmente proposta por EFSTRATIOU (2012). Seu trabalho aborda metodologias e vantagens do uso de RSASF compartilhadas para várias aplicações. LEONTIADIS & EFSTRATIOU (2012) propõem uma solução para a transformação de uma rede de sensores e atuadores tradicional em uma RSAC. O *SenShare* como é conhecido, oferece suporte a múltiplas aplicações e faz uma separação entre a infraestrutura e as aplicações. Cada aplicação opera de forma isolada como uma sub-rede abstrata dedicada. O gerenciamento das aplicações é efetuado por um nó sorvedouro dotado do sistema operacional *TinyOS (*DWIVEDI, 2009*)* e suas funcionalidades são divididas em camadas. O sorvedouro também é responsável por escalonar as tarefas dos atuadores em cada aplicação. O *SenShare* não suporta um número crescente de aplicações dadas as limitações computacionais existentes no sorvedouro. Dessa forma, podemos concluir que o *SenShare* não suporta uma alta escalabilidade de aplicações e não se preocupa com o gerenciamento por múltiplos usuários, pois as aplicações são criadas e executadas apenas no nó sorvedouro.

Além disso, no trabalho não é mencionado como os dados são armazenados ou tratados, ou mesmo se são guardados para consultas futuras.

FARIAS *et al*. (2013) fornece um solução voltada para a eficiência energética de RSACs utilizando como base o *SenShare* (LEONTIADIS & EFSTRATIOU (2012). Nessa solução, as aplicações com o menor número de dependências são executadas primeiro. As tarefas executadas por atuadores são escalonadas de acordo com a dependência de dados. Para entendermos o funcionamento da solução proposta por FARIAS, podemos fazer uma analogia tomando como exemplo uma rede que possua sensores de temperatura, umidade, presença e fumaça. As tarefas periódicas que necessitam de um menor grupo de dados como detecção de presença e incêndio são executadas primeiro. Os dados coletados são armazenados e utilizados para aplicações e tarefas que serão executadas posteriormente. Essa solução apresenta vantagens em relação ao *SenShare* pois oferece uma relação entre as aplicações e tarefas enquanto o *SenShare* trata as aplicações e tarefas separadamente em redes sobrepostas. Mesmo que essa solução forneça uma maior robustez à rede e seja capaz de fornecer uma melhoria na eficiência energética, essa solução herda os problemas de escalabilidade existente no *SenShare*.

ALANRI (2013) apresenta o *Sensor-Cloud*, um novo paradigma para a Computação em Nuvem que armazena dados de sensores e os compartilha através de uma infrestrutura de nuvem. A proposta do *Sensor-Cloud* é tratar os dados de sensores de forma mais eficiente e oferecer a possibilidade de desenvolvimento de múltiplas aplicações de monitoramento. Segundo MICROSTRAINS'S (2014) *Sensor-Cloud* é um serviço de armazenamento exclusivo para dados oriundos de sensores, visualização e plataforma de gerenciamento remoto, com o objetivo de oferecer escalabilidade de dados, visualização rápida e análise programável pelo usuário. Ele é originalmente projetado para suportar implementação de redes de sensores de longa duração. Recentes avanços tornaram o *Sensor-Cloud* capaz de suportar qualquer dispositivo conectado à web, sensores, ou rede de sensores através de uma API. No entanto, o *Sensor-Cloud* mesmo apresentando a reutilização dos dados coletados, ainda não apresenta uma solução para o reaproveitamento de infraestrutura de RSASF em mais de uma aplicação.

3 OBJETIVOS

3.1 Objetivo Geral

Fornecer uma solução para o gerenciamento de múltiplas aplicações utilizando redes de sensores e atuadores compartilhados (RSAC) como fonte de dados. Essa solução abordará alguns desafios encontrados no gerenciamento de redes de sensores como flexibilidade, disponibilidade dos dados coletados e escalabilidade de aplicações utilizando a Computação em Nuvem.

3.2 Objetivos Específicos

- Disponibilizar o compartilhamento e o gerenciamento de sensores e atuadores entre diversas aplicações;
- Fornecer o armazenamento em massa dos dados coletados em RSASF;
- Oferecer a disponibilidade de dados coletados por RSASF e;
- Fornecer uma ferramenta capaz de criar e gerenciar múltiplas aplicações através de uma interface Web de fácil entendimento e utilização.

4 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão abordados os principais conceitos, características e componentes de redes de sensores e atuadores. Além disso, serão descritos os modelos de armazenamento de dados de redes de sensores utilizados, os desafios e as propostas existentes. Por fim abordaremos a metodologia atualmente utilizada na integração de redes de sensores em fio com a Computação em Nuvem.

4.1 Redes de Sensores e Atuadores Compartilhados

RSAC é uma recente abordagem de redes de sensores, dada como uma evolução das redes de sensores tradicionais que oferece vantagens como a reutilização da infraestrutura e o desenvolvimento de múltiplas aplicações que podem ser gerenciadas por múltiplos usuários (BHATTACHARYA *et al*, 2010). A estrutura de uma RSAC é basicamente formada por uma RSSF que possui atuadores. No entanto, para entendermos a relação entre uma RSAC e uma rede de sensores sem fio (RSSF) precisamos analisar as características de uma RSSF.

4.1.1 Redes de Sensores sem Fio

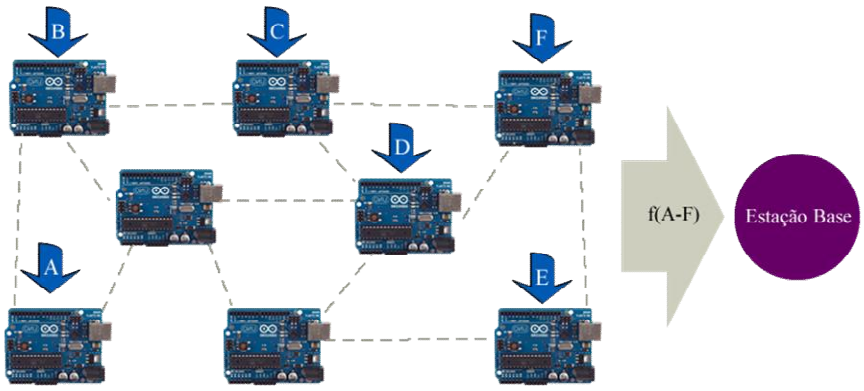
As RSSF têm recebido cada vez mais atenção devido a evolução de sistemas micro-eletrônicos que facilitam o desenvolvimento de sensores inteligentes. Tais sensores possuem baixo poder de processamento de dados e um tamanho reduzido, porém, recentes avanços na tecnologia nos levam a crer que as capacidades de processamento e memória são restrições temporárias em RSSFs (ROCHA, 2012). Esses sensores são capazes de detectar, medir e coletar informações do meio ambiente, e com base em algum processo de decisão local eles podem transmitir os dados monitorados para uma rede de computadores. (YICK, MUKHERJEE, & GHOSAL, 2008; ROCHA, 2012).

a) Características das RSSF

As características das RSSF podem variar de acordo com fatores como a área de interesse, requisitos das aplicações dentre outras. As RSSF possuem diversas características que podem diferenciá-las das demais redes como:

- os sensores podem possuir um endereço único, por exemplo, quando se deseja saber a localização exata do dispositivo. Por outro lado, em uma rede de sensores que tem o objetivo de monitorar uma determinada região, os dispositivos podem não precisar de um endereçamento único (YE & PAN, 2009);
- a rede é capaz de agregar ou sumarizar os dados coletados (figura 1). Essa funcionalidade pode ser útil para diminuir o número de mensagens enviadas pela rede (KSENTINI, 2014);

Figura 1 - Agregação de dados coletados pelos sensores



Fonte: elaborada pelo autor

- os sensores possuem localização fixa ou não. Por exemplo, sensores utilizados para controle de centrais de ar condicionados são tipicamente fixos, enquanto sensores utilizados para medir o nível da água de um rio ou oceano são móveis (YICK, MUKHERJEE, & GHOSAL, 2008);
- os sensores podem possuir restrição de intervalo para o envio dos dados coletados para uma entidade observadora;
- RSSFs podem possuir um número significativo de nós. Por exemplo, em redes de monitoramento de oceanos (VIEIRA; LOUREIRO, 2010) e florestas, o número de nós pode alcançar cem mil (YICK, J. 2008);
- a topologia da RSSF é dinâmica. Assim, nós sensores podem ficar inativos por diversos motivos como destruição física, falta de energia ou por decisão de algum algoritmo. Por outro lado, nós inativos podem se tornar ativos, ou novos nós podem tomar posse da posição de outro inativo. Em todo caso, torna-se necessário a utilização de um mecanismo de auto-organização para que a rede como um todo esteja sempre em funcionamento. Esse mecanismo deve ser automático e periódico;
- RSSF geralmente têm como objetivo alguma tarefa colaborativa, na qual é importante detectar e estimar eventos de interesse. Como as RSSF possuem algumas restrições de recursos, os dados podem ser sumarizados ou calculados para melhoria do processo de desempenho. Esse processo de sumarização depende da aplicação e;

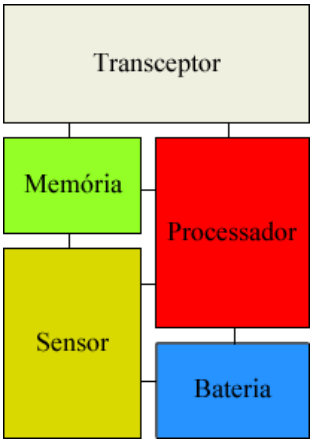
- um nó específico ou um grupo de nós podem ser utilizados para realizar consultas de dados coletados em uma determinada região de interesse. Dependendo do grau de sumarização executado, pode não ser viável transmitir os dados através da rede até o nó sorvedouro. Assim, é possível que vários nós sejam definidos como sorvedouros, que são responsáveis por coletar os dados e responder a consultas referentes aos nós sob sua "jurisdição".

b) Componentes de uma RSSF

Essa subseção descreve os principais elementos que compõem uma RSSF. Os principais componentes das RSSFs são os nós sensores, interfaces de comunicação sem fio e nós para se comunicar com outras entidades também conhecidos como nós *gateway* (DIALLO, 2012).

Os nós sensores são dispositivos independentes capazes de efetuar sensoriamento, processamento e comunicação. Os componentes básicos de um nó sensor são: transceptor, bateria ou fonte de energia, sensores, memória e processador (MACEDO; ARAÚJO; SOUSA, 2014). Brevemente, o transceptor é responsável pela comunicação sem fio. O processador responde pela capacidade de processamento do dispositivo. A memória é responsável pela execução de programas. Os nós possuem uma memória interna, mas podem receber memórias externas acopladas ao dispositivo. O sensor é responsável por detectar e coletar os dados do ambiente. A bateria ou fonte de alimentação é responsável por alimentar o dispositivo, quando há o uso de baterias, a vida útil do dispositivo está diretamente ligada à quantidade de energia da bateria utilizada. Uma ilustração dos componentes de uma RSSF pode ser visualizada na figura 2.

Figura 2 - Estrutura física básica de um nó sensor



Fonte: elaborado pelo autor.

c) Redes de Sensores e Atuadores

Como mencionado anteriormente, uma RSASF é composta de uma RSSF que possui atuadores. Atuador é um dispositivo capaz de executar tarefas de automação, tarefas programadas e responder a comandos remotos (Redes Industriais para sensores e atuadores, 2013). Muitos trabalhos tratam sensores e atuadores como sendo o mesmo tipo de dispositivo, isso ocorre comumente pelo fato de que nós atuadores também podem ser sensores (que sensoriam o ambiente, coletando informações a respeito dele) e são comuns em RSASF. Entretanto, é necessário fazer uma distinção entre nó atuador e simplesmente dispositivos atuadores. Nesse trabalho vamos tratá-los de forma separada, já que funções de sensoriamento e atuação serão utilizadas com objetivos diferentes.

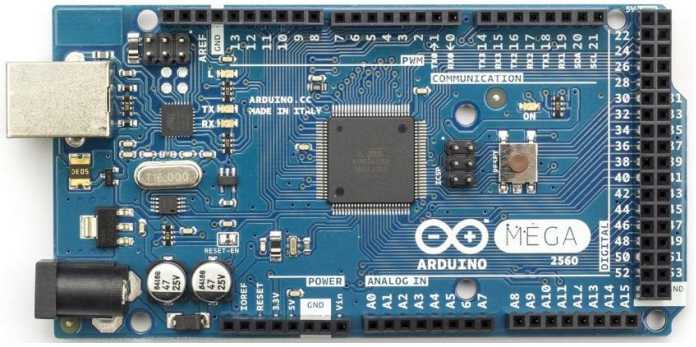
4.2 Plataforma Arduino

Arduino é uma plataforma para o desenvolvimento de protótipos com *hardware* livre e de fácil implementação. O público alvo da plataforma arduino é basicamente o usuário que possua pouco ou nenhum conhecimento de eletrônica e programação (McRoberts, 2011, p. 22).

A estrutura física da plataforma Arduino é composta por um microcontrolador central e portas de entrada e saída (Figura 3). Os programas desenvolvidos para plataforma Arduino

são escritos nas linguagens C e C++. Um ambiente de desenvolvimento integrado (*Integrated Development Environment* – IDE) é oferecido para o desenvolvimento de softwares para Arduino (ARDUINO S.A., 2014). A IDE Arduino apresenta funcionalidades como o envio do programa criado para o microcontrolador, a verificação de sintaxe e semântica do código criado, monitoramento das portas seriais e fornece suporte a todos os modelos de microcontroladores utilizados em plataformas Arduino.

Figura 3- Plataforma de prototipagem Arduino modelo Mega 2560 R3



Fonte: Arduino S.A. (2014).

O microcontrolador utilizado na plataforma é dotado de portas analógicas que suportam a entrada de dados de sensores e portas digitais que podem ser de entrada ou saída. As portas digitais geralmente são responsáveis por acionar dispositivos atuadores ou detectar mudanças discretas na entrada de dados (MELLIS, 2009). Para aumentar as funcionalidades da plataforma existem complementos conhecidos como *Shields*. Esses complementos possuem funcionalidades específicas como comunicação sem fio, conjuntos de sensores, comunicação GSM, comunicação via satélite e serviços de rede. Os *Shields* se comunicam com o microcontrolador através de portas seriais.

4.2.1 Módulo de Comunicação Xbee

O módulo Xbee é um dispositivo capaz de executar a tarefa de comunicação entre dispositivos via rádio frequência. O Xbee pode ser integrado facilmente à plataforma Arduino como um *Shield,* adicionando assim a funcionalidade de comunicação sem fio à plataforma. O módulo Xbee pode ser encontrado em algumas versões sendo que duas são as mais comumente utilizadas: o XBee e o XBee-PRO (Figura 4). Ambas versões oferecem as

mesmas funcionalidades, a diferença está apenas na potência de transmissão (ESCHNER, 2011).

Figura 4 - Módulos Xbee e Xbee-PRO

Fonte: Digi Internacional (2014).

O Xbee utiliza o protocolo ZigBee, destinado a aplicações com baixo consumo de energia e baixa transferência de dados (CUNHA, 2013). O protocolo ZigBee foi desenvolvido para ser utilizado em diversos tipos de topologias de RSSFs. O uso desse protocolo é muito comum em redes de sensores pois oferece confiabilidade e maior alcance (SALEIRO, 2014).

4.3 Gerenciamento de dados de RSASF

Em RSASFs, os dados coletados são enviados para a internet utilizando *gateways*, os usuários podem acessar os dados através de consultas (DIALLO, 2012). Uma vez que os dados possam ser consultados, surge também a necessidade de armazenamento. Dependendo dos tipos de dados e da frequência das consultas, pode surgir a necessidade de armazenamento em massa dos dados.

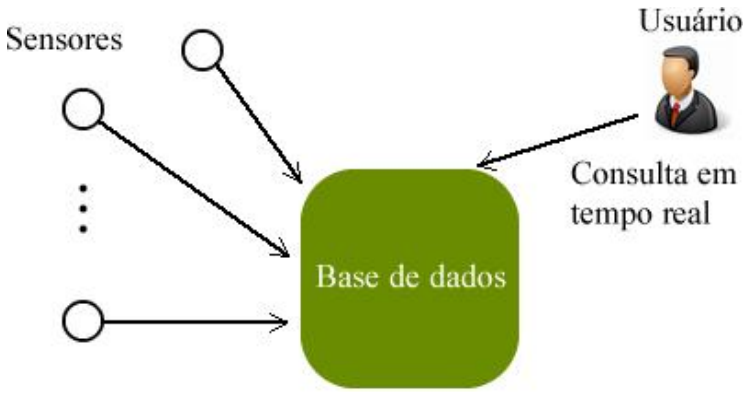
Os dados coletados por uma RSASF devem refletir o estado atual do ambiente alvo da forma mais precisa possível. No entanto, esse ambiente pode mudar constantemente e os dados podem ser coletados em períodos discretos. Assim, a eficácia dos dados possui valor temporário, que com o passar do tempo se tornam mais ou menos precisos, até se refletirem no estado atual do ambiente. O principal objetivo do gerenciamento de RSASF é garantir a consistência de dados temporais e operações de processamento dentro de restrições de tempo real (DIALLO, 2012).

4.3.1 Modelos de Armazenamento de dados em redes de sensores

Segundo DIALLO (2012) existem dois tipos de abordagem para o armazenamento de dados de uma rede de sensores:

- Armazenamento centralizado (Figura 5): os dados coletados são salvos em um banco de dados central, cujas consultas dos usuários são processadas. Essa abordagem é uma das mais utilizadas e possui problemas como: gargalo no servidor de consultas devido ao aumento da quantidade de dados; e existência de ponto (servidor) único de falha e sobrecarga na rede. Esse tipo de abordagem pode oferecer atrasos no tempo de consultas, tornando-a não recomendada para sistemas de gerenciamento em tempo real.

Figura 5 - Abordagem de armazenamento centralizado

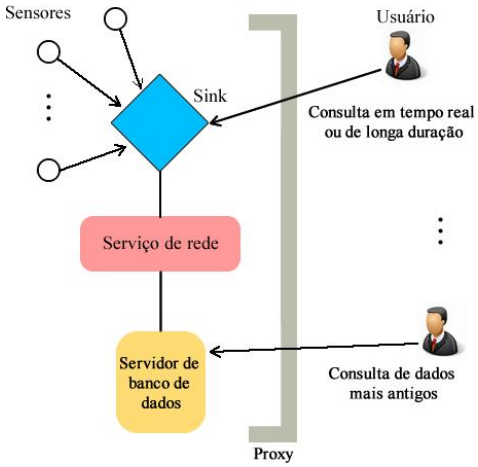


Fonte: elaborada pelo autor.

- Armazenamento distribuído (Figura 6), explora as capacidades de cálculo e armazenamento dos nós sensores. Para aumentar a economia de energia na rede são efetuados cálculos localmente antes do envio. Nessa abordagem, os dados são salvos em um servidor e parcialmente na própria rede. Dessa forma, os dispositivos atuam como parte da base de dados. Essa abordagem pode oferecer vantagens como o processamento de consultas realizadas na rede em tempo real e consultas de dados mais antigos.

Uma abordagem de armazenamento distribuído apresenta vantagens significativas sobre outras abordagens. Recentes estudos na área de gerenciamento de dados de RSSFs têm contribuído para o avanço de técnicas de armazenamento distribuído utilizando a Computação em Nuvem, como é o caso do *Sensor-Cloud* (ALANRI, 2013).

Figura 6 - Abordagem de armazenamento de dados distribuída.



Fonte: elaborada pelo autor.

4.4 Integração de RSASF com a Computação em Nuvem

Nessa subseção serão abordados alguns dos métodos recentemente utilizados para a integração de RSASFs com a Computação em Nuvem. Para o desenvolvimento de uma solução capaz de executar aplicações com dados oriundos de RSASF é necessário conhecer as características da Computação em Nuvem, seu funcionamento e por fim as metodologias de integração com redes de sensores.

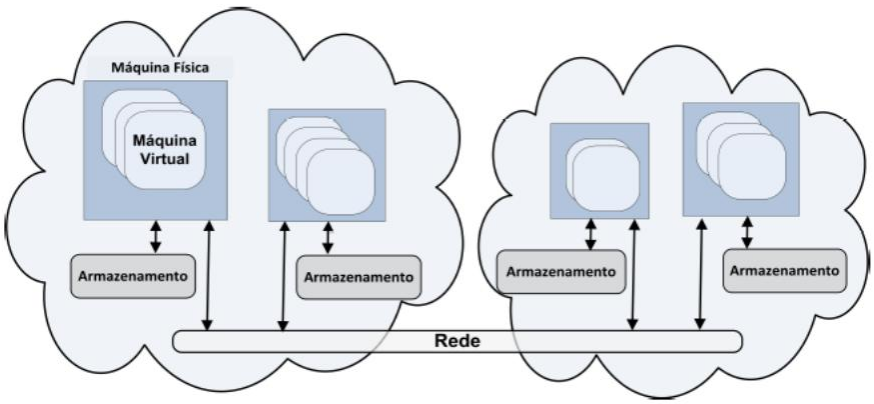
4.4.1 Computação em Nuvem

Segundo AROCKIAM (2011) a computação em nuvem é uma tecnologia de computação que utiliza a internet e vários servidores remotos para manter dados e aplicativos de software. A Computação em Nuvem permite aos usuários utilizar aplicações de *software* robustos sem precisar instalá-los no computador local.

Nos dias atuais, serviços como abastecimento de energia, água, transporte e combustíveis são oferecidos livremente aos consumidores. Tais serviços são cobrados de acordo com o uso, ou seja, paga-se pela quantidade que utiliza-se tal serviço. A Computação em Nuvem tem como objetivo algo semelhante, oferecer serviços de tecnologia com custo baseado no uso. O usuário final paga apenas uma taxa pela utilização do serviço na nuvem. O serviço é oferecido diretamente nos servidores dos provedores e é acessado pelo usuário final através da internet. Os recursos da nuvem são monitorados a fim de possibilitar a tarifação de acordo com o uso de cada recurso da mesma. Consequentemente, o usuário pode optar por adquirir um serviço sem a necessidade de investir em equipamentos.

A arquitetura da nuvem é geralmente formada por uma grande quantidade de servidores conectados através de uma rede. Os servidores geralmente utilizam os mesmos recursos de *software,* mas podem possuir capacidades de *hardware* como CPU, memória e armazenamento diferentes. Para um melhor aproveitamento dos recursos de cada computador, cada máquina possui um determinado número de máquinas virtuais executando interiormente, de acordo com a capacidade de *hardware* da máquina física (SOUZA *et al*, 2011). A arquitetura ainda oferece uma interface *front-end* como um Portal que permite ao usuário escolher o serviço desejado como um catálogo.

Figura 7 - Ambiente da Computação em Nuvem



Fonte: SOUZA *et al*. 2011

4.4.2 Modelos de Serviço

No atendimento a múltiplos usuários é comum haver uma grande distinção entre as necessidades dos mesmos, tornando essencial a capacidade de personalização dos recursos da nuvem. Desde serviços de infraestrutura, a serviços de plataforma e serviços de *software* (AROCKIAM, 2011). Nesse contexto, a Computação em Nuvem oferece três modelos de serviços: software como um serviço (SaaS), plataforma como um serviço (PaaS) e infraestrutura como um serviço (IaaS) (AROCKIAM, 2014). Tais modelos são definidos a seguir:

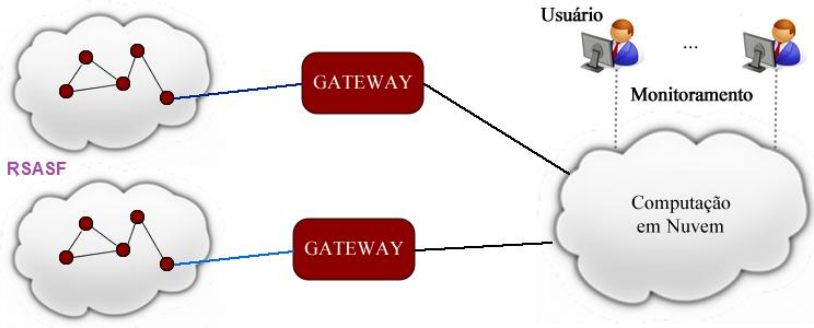
- Infraestrutura como um Serviço (IaaS): a nuvem oferece para os usuários recursos de computação como armazenamento, servidores, rede e outros recursos. Em vez de comprar esses recursos, os usuários apenas precisam pagar pelo uso dos mesmos. Os usuários ainda têm o direito de controlar quanto utilizar de armazenamento, sistemas operacionais, aplicativos implementados e controle limitado de componentes de rede selecionados. Exemplo: *Amazon Elastic Cloud Computing* (EC2).
- Plataforma como um Serviço (PaaS): a nuvem fornece uma plataforma que possibilita os usuários implementarem aplicativos e *softwares* que podem ser acessados de qualquer lugar através da internet. Os recursos utilizados por aplicações desenvolvidas na nuvem geralmente aumentam e diminuem de acordo com o uso. Esse modelo de serviço, na maioria das vezes, evita que licenças de *software* muito caras e ou complexos ambientes de desenvolvimento sejam adquiridos pelos desenvolvedores. É necessário apenas o pagamento pelo período de desenvolvimento das aplicações. Exemplo: *Google App Engine* e *Windows Azure*.
- Software como um Serviço (SaaS): a nuvem oferece aplicativos e *softwares* para os usuários. Não há necessidade de instalação de nenhum serviço de *software* localmente para a utilização desse modelo de serviço. O requisito mínimo para o acesso a todos os serviços em nuvem é apenas um computador e conexão com a internet. O usuário final poderá consumir os recursos do software mas não têm o direito de controlar ou gerenciar a infraestrutura da nuvem. Exemplo: *Google docs*.

4.5 Método de Integração de RSASF e Computação em Nuvem

Recentes trabalhos buscam oferecer soluções para integração de RSASF e Computação em Nuvem visando confiabilidade dos dados, disponibilidade e um melhor aproveitamento dos dados coletados. Em suma, os dados coletados são enviados para os gateways que devem estar conectados com a nuvem (BRITO, 2014). Os dados armazenados

são processados e podem ser monitorados a qualquer momento por múltiplos usuários. O processamento das aplicações passa a ser realizado na nuvem (figura 8).

Figura 8 - Modelo de integração de RSASF e a Computação em Nuvem.



Fonte: elaborada pelo autor.

Uma vez que os dados coletados estão na nuvem, os mesmos podem ser oferecidos como fonte para o desenvolvimento de aplicações ou como SaaS na nuvem (SHAH, 2013). As políticas de acessos aos dados são realizadas na nuvem, o que torna necessário o desenvolvimento de uma solução capaz de controlar acessos, escalonar tarefas e garantir a disponibilidade dos dados coletados.

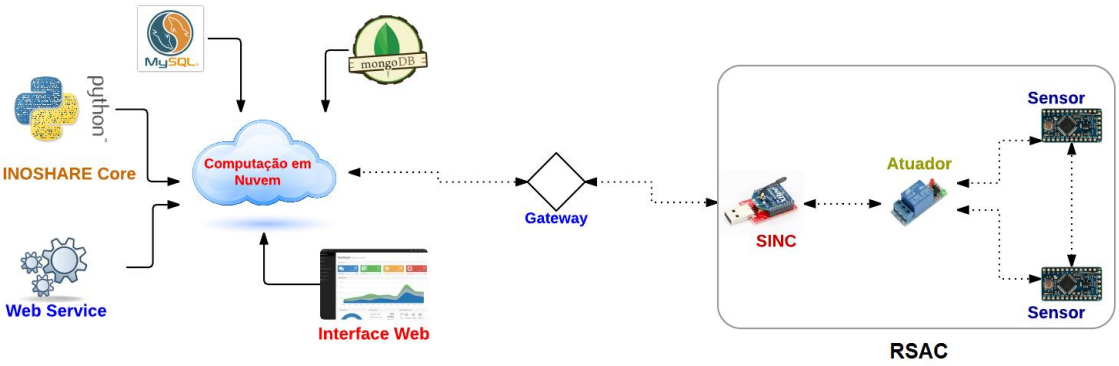
5 INOSHARE

Nas próximas seções serão descritos os principais componentes do INOSHARE e seu respectivo funcionamento.

5.1 Descrição geral

O INOSHARE visa prover a reutilização da infraestrutura de redes de sensores e atuadores por mais de uma aplicação. As aplicações criadas possuem condições que quando verdadeiras podem gerar comandos para a rede. Essa característica pode ser chamada de aplicação de condição e ação (RIVERO; DOORN; FERRAGINE, 2006). Os principais componentes do INOSHARE são a RSSF e os serviços que se encontram na nuvem. A figura 9 mostra com mais detalhes a estrutura geral do INOSHARE. Ao disponibilizar a reutilização dos recursos de uma rede de sensores e atuadores, teremos o que pode ser denominado como rede de sensores e atuadores compartilhados (RSAC) (LEONTIADIS & EFSTRATIOU, 2012).

Figura 9 - Estrutura lógica do INOSHARE



Fonte: Elaborada pelo autor.

O papel da RSASF nessa abordagem é simplesmente coletar dados e executar comandos originários das aplicações que serão executadas na nuvem. O fluxo de funcionamento se inicia no envio de dados coletados para o servidor de banco de dados. As aplicações são criadas com suas informações salvas no servidor de banco de dados relacional.

O que chamamos nesse trabalho de INOSHARE *Core* é o responsável por ler e executar as aplicações salvas no banco relacional. O *Web Service* é o responsável por prover dados coletados pela RSSF existentes na base de dados não relacional, que podem ser utilizados em outras aplicações. A interface Web é um simples meio de criar e acompanhar as execuções das aplicações cadastradas pelo usuário.

5.2 Protótipo

5.2.1 A rede de sensores

A RSASF deve ser composta de nós sensores e atuadores e pelo menos um nó sink (coordenador) responsável por receber os dados e enviar para os *gateways* (deve existir pelo menos um) existentes na rede.

Os dados enviados pela RSASF seguem o formato JSON que é idêntico ao formato de dados do serviço de bancos de dados não relacional *mongodb* (MONGODB.INC, 2014). Em caso de utilizar outra interpretação como XML, será necessário desenvolver um meio de converter os formatos.

5.2.2 O gateway

A funcionalidade dos *gateways* é receber todos os dados coletados pela rede de sensores e enviá-los para a rede externa (YICK, 2008). Nesse trabalho os *gateways* se conectam com a nuvem e têm o papel de enviar e receber dados. Quando a nuvem processa as aplicações e envia os comandos a serem executados pela rede, são os *gateways* quem os recebem e enviam para a rede de sensores. Em paralelo com isso, os dispositivos *gateway*s recebem os dados coletados e os enviam para o servidor de banco de dados *mongodb* contido na nuvem.

5.2.3 A nuvem

A infraestrutura de nuvem configurada para esse projeto foi uma máquina virtual, com sistema operacional *Red Hat* (Hat.INC, 2014) contendo 2048MB de memória RAM, CPU com 2.5 GHz e armazenamento SSD de 30GB. No entanto, configurações menos robustas podem ser utilizadas para executar um número pequeno de aplicações, como poderá ser visto nos resultados da seção 7.1. Na instância são centralizados os serviço de bases de dados, *Core* do INOSHARE, Interface web e *Web service*, tendo respectivamente os serviços:

- servidor SSH;
- servidor de banco de dados *mongodb*;
- servidor de banco de dados relacional MySQL; e,
- servidor web apache e interpretador PHP.

Tabela 1 - Portas e protocolos abertos para entrada na instância da nuvem.

Número da porta	**Protocolo**	**Aplicação**
27017	TCP	*mongodb*
22	TCP	*SSH*
80	HTTP	*Apache*
3306	TCP	*MySQL*
445	ICMP/UDP	*Ping*

5.2.4 As bases de dados

a) Banco de dados MongoDB

O formato de dados utilizado pelo mongodb é conhecido como chave-valor, que se assemelha com o modelo relacional (HAN, 2011). O modelo de armazenamento do *mongodb* é realizado por documentos. Fazendo uma analogia, no *mongodb* o que é chamado de documento é conhecido como tabela no modelo relacional,. Em bases de dados NoSQL, tudo que é inserido possui uma chave única que recebe o nome de "_id" (MONGODB.INC, 2014). A estrutura dos dados salvos no banco de dados, deve ser suficiente para gerar as aplicações e possui a seguinte aparência: {"_id":{"$id":"5435c0b6e7b9916e02ede621"},"lab":"2","porta":"A0","id_no":1,"datahora":{ "sec":1411568798,"usec":0},"valor":"828","sensor":"1"}. Onde:

- "_id" é gerado automaticamente pela base de dados;
- "lab" é o nome dado a localização do dispositivo físico na rede (no caso, campus da universidade federal do ceará (UFC), seguindo a mesma ideia do ClimaDuino);
- "porta" indica a localização do sensor na plataforma Arduino;
- "id_no" é um número sequencial para endereçar ou identificar os nós na rede;
- "datahora" representa a data e hora em que os dados foram enviados para o banco de dados;

- "valor" é o valor coletado pelo sensor; e,
- "sensor" é um número único que identifica o sensor inserido no Arduino.

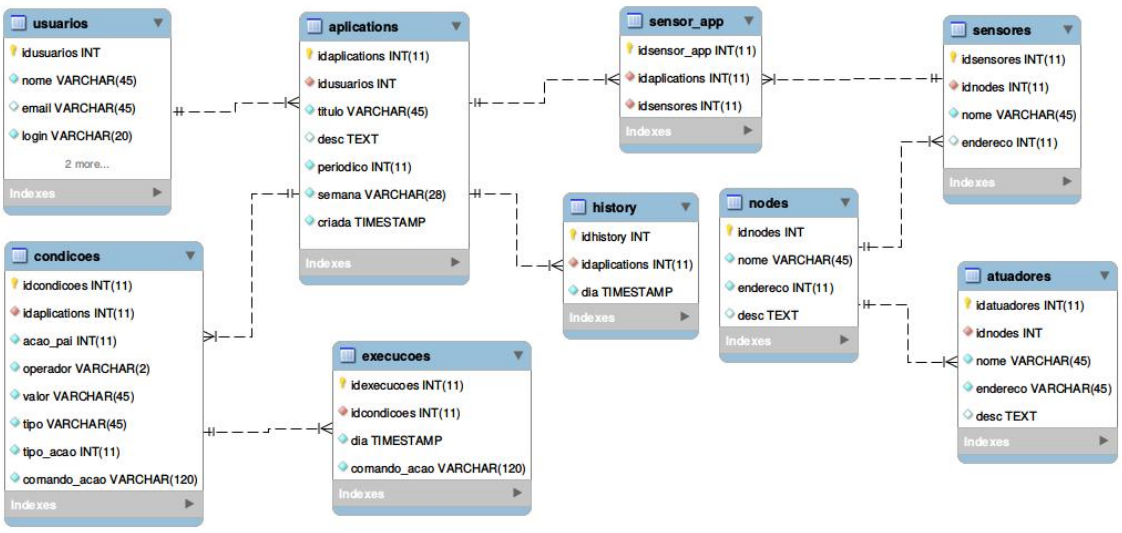
b) Banco de dados MySQL

O serviço de banco de dados MySQL é um sistema de gerenciamento de banco de dados relacional, o qual utiliza como linguagem o SQL (ORACLE, 2014). Nesse projeto, as aplicações criadas pelo usuários, tem os seus dados armazenados em um banco de dados MySQL. Na figura 10 é apresentado o diagrama de Entidade Relacionamento (ER) do banco de dados utilizado pelo INOSHARE. Nessa subseção serão abordados os detalhes do banco de dados do INOSHARE e os detalhes superficiais da estrutura de dados das tabelas mais significativas para o funcionamento do sistema.

c) Tabela usuarios

Na tabela usuários ficam armazenados os dados dos usuários do sistema que pretendem criar aplicações a partir da interface web do INOSHARE. Não é permitido que uma aplicação seja criada sem que haja uma referência a um usuário já cadastrado no banco de dados.

Figura 10 - diagrama de entidade relacionamento do banco de dados MySQL.



Fonte: elaborada pelo autor

d) Tabela aplications

Na tabela "aplications" são armazenados os dados primários das aplicações. Na figura 11 podemos observar com mais detalhes o formato e o armazenamento dos dados.

Figura 11 - a tabela "aplications"

idaplications	idusuarios	titulo	desc	periodico	semana	criada
1	1 [->]	Controle de Ar Condicionado	GestÃ£o de aparelhos de Condicionamento de Ar no Campus QuixadÃ¡.	60	dom;seg;ter;qua;qui;sex;sab	2014-11-14 10:22:25
2	2 [->]	Deteccao de Intruso	Detecta a presenca de intrusos no fim de semana	30	dom;sab	2014-10-23 00:48:30
3	1 [->]	Risco de Fogo	Detecta uma ameaÃ§a de incÃªndio.	60	dom;seg;ter;qua;qui;sex;sab	2014-11-16 18:32:07

Fonte: elaborada pelo autor

A coluna "idaplications" é a chave primária da tabela e possui valor automaticamente incrementado. Essa coluna serve como referência para outras tabelas como a de histórico de execuções e a tabela de condições das aplicações. Na coluna "periodico" é armazenado o valor em segundos que a aplicação deverá aguardar até ser executada novamente. Em seguida, temos a coluna "semana" que deve possui uma abreviação dos dias da semana que a aplicação deve ser executada. Por fim, na coluna "criada" tempos o *timestamp* do momento em que a aplicação foi cadastrada no banco de dados.

e) Tabela condicoes

A tabela denominada "condicoes" possuem as condições para execução de comandos existentes. A figura 12 apresenta a estrutura da tabela e os dados das condições que foram inseridas nesse projeto para a execução das aplicações cadastradas.

Figura 12 - detalhes da tabela "condicoes"

idcondicoes	idaplications	tipo_coleta	end_sensor	acao_pai	operador	valor	tipo	tipo_acao	comando_acao
1	1 [->]	2	2;4	0	>	0	INT	0	3;1;1;1;4;x
2	1 [->]	2	1;3	1	<	20.0	FLOAT	0	3;1;1;1;2;x
3	1 [->]	2	1;3	1	>	23.0	FLOAT	0	3;1;1;1;3;x
4	1 [->]	2	2;4	0	<=	0	INT	0	3;1;1;1;5;x
5	2 [->]	2	5;6	0	>	0	INT	1	jcarlosss.carvalho@gmail.com
6	3 [->]	2	8;9	0	>	30.00	DOUBLE	0	
7	3 [->]	2	10;11	6	<	40.00	DOUBLE	1	ccarvalho@alu.ufc.br

Fonte: elaborada pelo autor

Na tabela "condicoes" as colunas "idaplications" e "end_sensor" são referencias da aplicação e dos sensores dos quais a condição se refere respectivamente. No INOSHARE é possível construir uma sequencia de condições aninhadas, onde na coluna "acao_pai" é inserido a relação de dependência dessa condição. As condições que não dependem de outras condições são caracterizadas com o valor zero na coluna "acao_pai". Uma condição que possui uma relação de dependencia só é testada se a sua condição primária for verdadeira. A coluna "operador" suporta os sinais: maior (>), menor (<), maior ou igual (>=), menor ou igual (<=), igual (==) e diferente (!=). A coluna "tipo_coleta" pode receber os valores: i) "1" para indicar valor único, que na prática se refere ao sistema comparar um único valor dos sensores com a coluna "valor"; ii) "2" para indicar média, que se refere que o valor a ser comparado será a média de valores coletados pelos sensores em questão, dentro do intervalo de tempo determinado na coluna "periodico" da tabela "aplications"; iii) "3" para indicar somatória, nessa opção o sistema irá utilizar como comparador a somatória dos dados coletados pelos sensores indicados na coluna "end_sensor". A coluna "tipo_acao" indica se o tipo de ação que será executada caso a condição seja verdadeira é do tipo comando para a rede (valor 0) ou envio de email (valor 1). Na coluna "comando_acao" são armazenados os comandos que devem ser enviados e ou os endereços de emails na qual uma mensagem deverá ser enviada caso a condição seja verdadeira.

A coluna "comando_acao" possui a seguinte lógica de preenchimento:

- os valores deverão ser separados por ponto e virgula;
- o primeiro valor indica o endereço do nó;
- o segundo valor indica o *id* do sensor ou atuador;
- o terceiro valor indica a porta que está inserido o sensor ou atuador;
- o quarto valor indica a ação que deve ser executada;
- o quinto valor indica o tipo de ação, que pode ser o valor zero para mudar o estado do dispositivo para nível lógico baixo, valor um para mudar o nível lógico para alto ou outro valor que deve ser previsto no código embarcado do nó, como por exemplo, nesse projeto o valor 2 indica aumentar a temperatura; e,
- o último valor é apenas para indicar o fim da mensagem, sendo enviado um caractere qualquer.

f) Tabela history e tabela execucoes

As tabelas "history" e "execucoes" são responsáveis por armazenar os históricos de execução de aplicação e condições verdadeiras testadas respectivamente. Com base na tabela "history" as aplicações são executadas, levando em consideração o período de tempo que cada aplicação deverá passar sem executar. A tabela "execucoes" armazena os históricos de condições verdadeiras que executaram alguma ação.

Figura 13 - detalhes da tabela "history"

idhistory	idaplications	dia
1	1 [->]	2014-09-24 07:45:32
2	3 [->]	2014-09-24 07:45:32
3	1 [->]	2014-09-24 07:46:32
4	3 [->]	2014-09-24 07:46:32
5	1 [->]	2014-09-24 07:47:32
6	3 [->]	2014-09-24 07:47:32
7	1 [->]	2014-09-24 07:48:32
8	3 [->]	2014-09-24 07:48:32
9	1 [->]	2014-09-24 07:49:32
10	3 [->]	2014-09-24 07:49:32

Fonte: elaborada pelo autor.

Figura 14 - detalhes da tabela "execucoes"

idexecucoes	idcondicoes	dia	comando_acao	valorsensor
31	1 [->]	2014-09-24 08:02:34	3;1;1;1;4;x	518
32	3 [->]	2014-09-24 08:02:34	3;1;1;1;3;x	27.0
33	1 [->]	2014-09-24 08:03:34	3;1;1;1;4;x	483
34	3 [->]	2014-09-24 08:03:34	3;1;1;1;3;x	26.0166666667
35	1 [->]	2014-09-24 08:04:34	3;1;1;1;4;x	515
36	3 [->]	2014-09-24 08:04:34	3;1;1;1;3;x	26.0
37	1 [->]	2014-09-24 08:05:34	3;1;1;1;4;x	477
38	3 [->]	2014-09-24 08:05:34	3;1;1;1;3;x	26.0
39	1 [->]	2014-09-24 08:06:34	3;1;1;1;4;x	405
40	3 [->]	2014-09-24 08:06:34	3;1;1;1;3;x	26.0

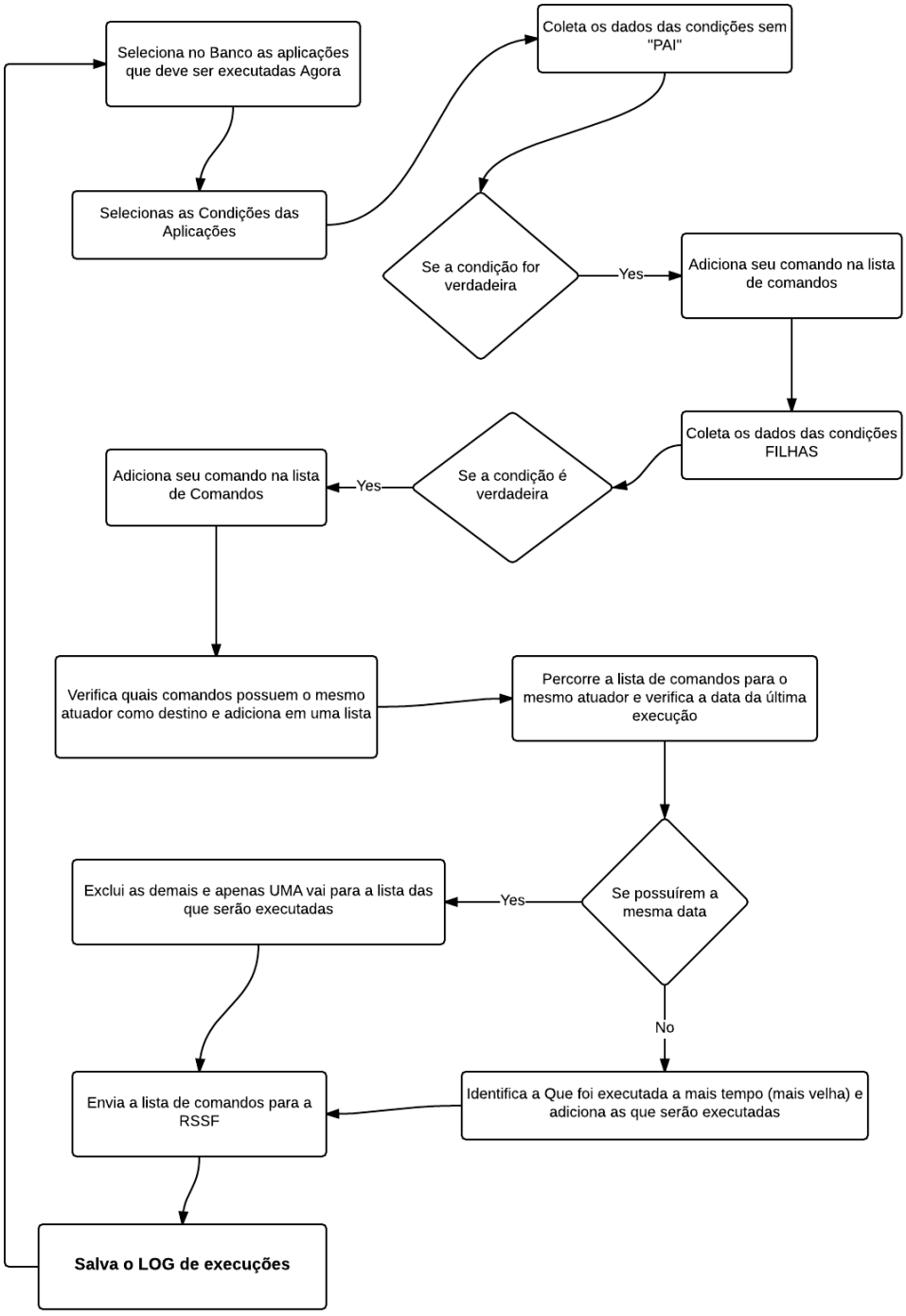
Fonte: elaborada pelo autor.

As tabelas "sensores", "atuadores" e "nodes" são responsáveis por armazenar informações sobre a rede como o id dos nós, endereços dos sensores e atuadores.

5.2.5 O INOSHARE Core

O módulo INOSHARE *Core* foi desenvolvido utilizando a linguagem *Python,* versão 2.8 (FOUNDATION, 2014). Ele é o responsável por executar as aplicações cadastradas no banco de dados. Seu funcionamento pode ser visualizado no fluxograma de processos na figura 15.

Figura 15 - diagrama de processos de execução do INOSHARE Core



Fonte: elaborada pelo autor.

Depois que são identificadas as aplicações que serão executadas naquele momento, as condições que não possuem relação de dependência (na figura 15 referenciadas como ações sem "PAI") serão testadas e se caso a condição for verdadeira, as suas condições dependentes (na figura 15 referenciadas como condições "FILHAS") também serão testadas. Para evitar

que dois ou mais comandos sejam enviados simultaneamente para o mesmo atuador, uma filtragem é realizada. Inicialmente é verificado se as condições verdadeiras para o mesmo atuador possuem a mesma data de última execução na tabela "execucoes", isso poderia acontecer se caso nenhuma das condições tenha sido executada ainda, em caso afirmativo somente uma condição será enviada, as demais serão excluídas. Em casos onde as condições possuem as datas de última execução diferentes, a condição que foi executada a mais tempo pela última vez em comparação com as demais será executada.

5.2.6 O Web Service

O módulo w*eb service* foi criado utilizando a linguagem de programação PHP na versão 5.4 (GROUP, 2014), sendo compatível com as versões maiores ou iguais a 5.3.2. A tarefa do módulo w*eb service* é prover através de requisições HTTP um conjunto de dados existentes no servidor de banco de dados *mongodb*.

O formato de retorno dos dados é JSON (VOGEL, 2014). A URL principal do serviço utilizado nesse projeto é "http://inoshare.com.br/api/web/". A tabela 2 apresenta informações sobre os parâmetros que podem ser utilizados, quais os respectivos retornos do servidor e exemplos de utilização. A figura 16 mostra um exemplo de retorno de requisição ao *web service*, no exemplo da figura a consulta retorna 10 resultados iniciando a exibição a partir do 50° registro em ordem decrescente.

Tabela 2 - detalhes de parâmetros de requisição do *web service*

Parâmetro	**Retorno**	**Exemplo de utilização**
Documento	Últimos 40 registros inseridos no banco de dados	http://inoshare.com.br/api/web/**inoshare**
documento/ qtd/N	N últimos registros inseridos no banco de dados.	http://inoshare.com.br/api/web/**inoshare /qtd/10**
documento/ qtd/N/offset /X	N registros inseridos no banco de dados até o registro X	http://inoshare.com.br/api/web/**inoshare /qtd/10/offset/15**

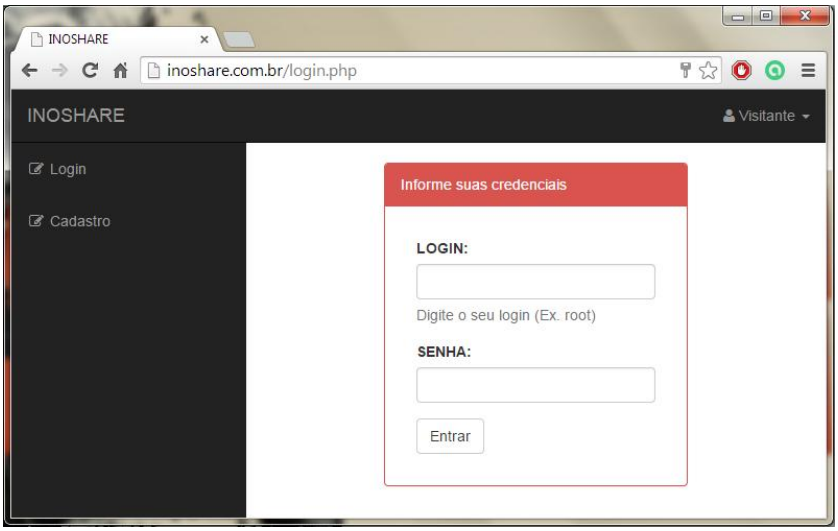
Figura 16 - exemplo de retorno de requisição ao web service

Fonte: elaborada pelo autor

5.2.7 A interface web

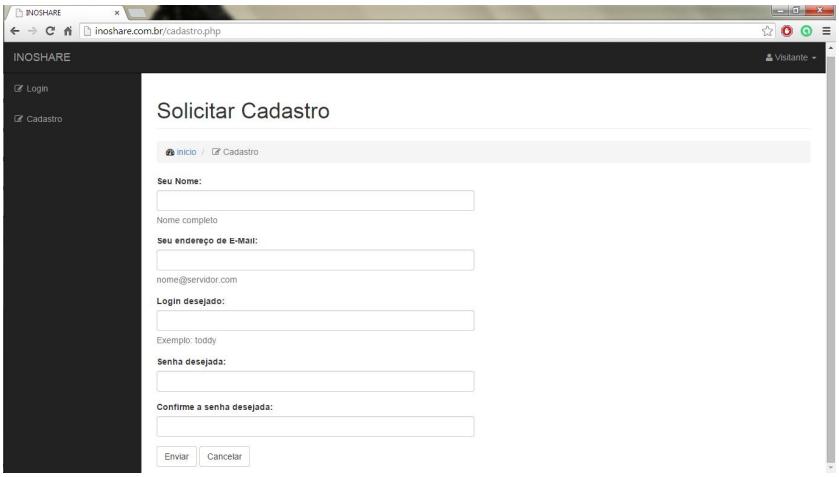
A interface web provê ao usuário um meio de criar aplicações e monitorar as execuções de comandos enviados para a RSAC. O primeiro passo para que o usuário crie suas aplicações é acessar o endereço da web do INOSHARE (http://inoshare.com.br). O usuário deverá fornecer um nome de *login* e uma senha existentes no banco de dados, por outro lado se não possuir cadastro poderá solicitá-lo clicando no link "Cadastro" (http://inoshare.com.br/cadastro.php). A solicitação de cadastro será enviada para o administrador via e-mail. O administrador do sistema deverá decidir se insere o usuário no sistema.

Figura 17 - Tela de login da Interface Web do INOSHARE.



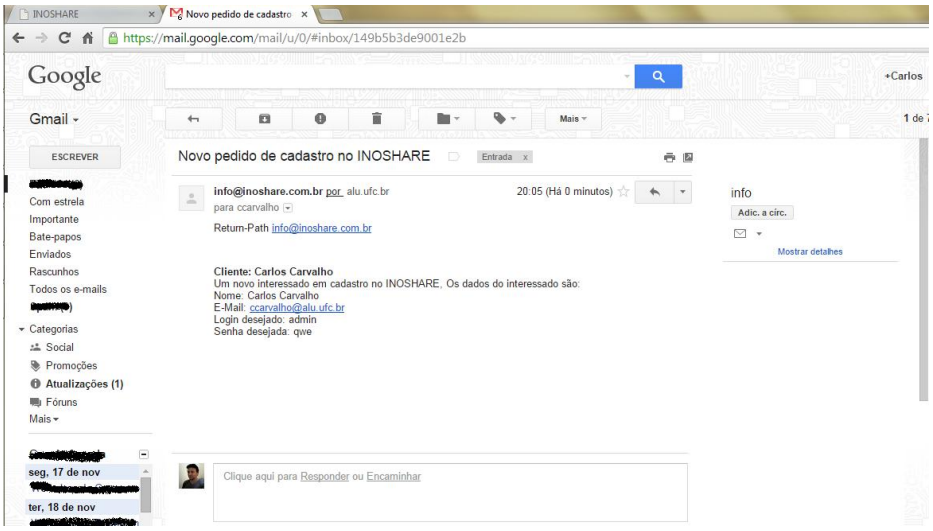
Fonte: elaborada pelo autor.

Figura 18 - Formulário de solicitação de cadastro no sistema INOSHARE



Fonte: elaborada pelo autor.

Figura 19 - Exemplo de e-mail de pedido de cadastro enviada a partir do formulário de solicitação (figura 22).

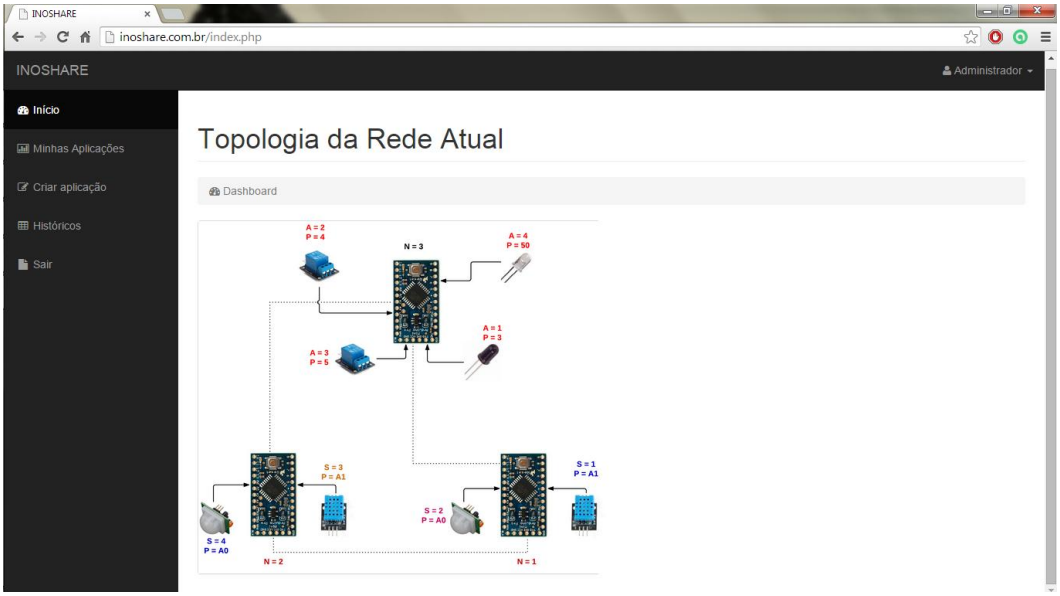


Fonte: elaborada pelo autor.

Após o usuário autenticar-se no sistema, a página inicial fornecerá uma visão geral da infraestrutura da RSAC que estiver conectada (Figura 20), que é composta apenas de uma figura fixada na página inicial. Nessa página o usuário poderá identificar: os endereços dos nós, a identificação das portas onde cada sensor e atuador estiver conectado e o número itens da infraestrutura da rede. O menu lateral da página principal da interface possui as opções: Início, Minhas Aplicações, Criar Aplicações, Históricos e Sair. O item Início leva o usuário a

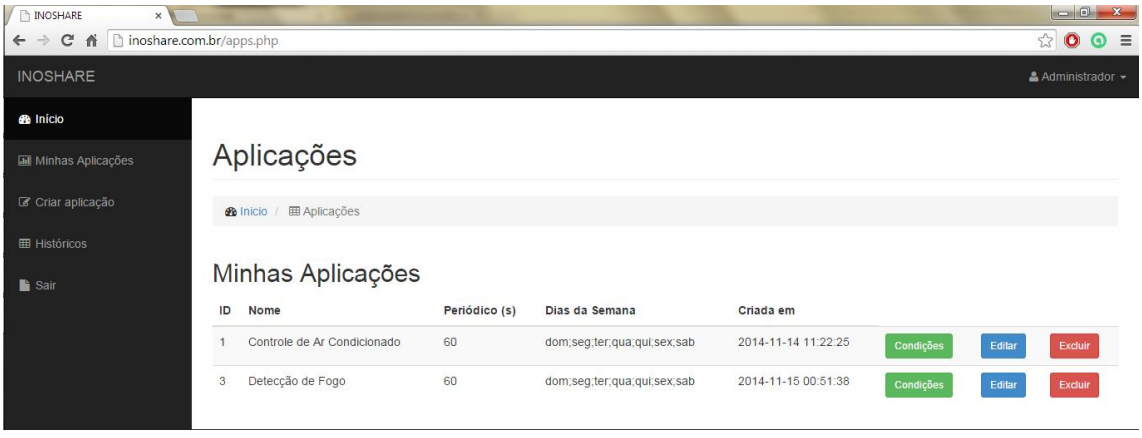
página inicial já citada. Na página "Minhas Aplicações" o usuário poderá visualizar uma lista de aplicações cadastradas por ele, contando ainda com a opção de ver a condições cadastradas para a aplicação, editar informações sobre a aplicação e removê-la completamente (Figura 21).

Figura 20 - Página inicial da interface Web do INOSHARE.



Fonte: elaborada pelo autor.

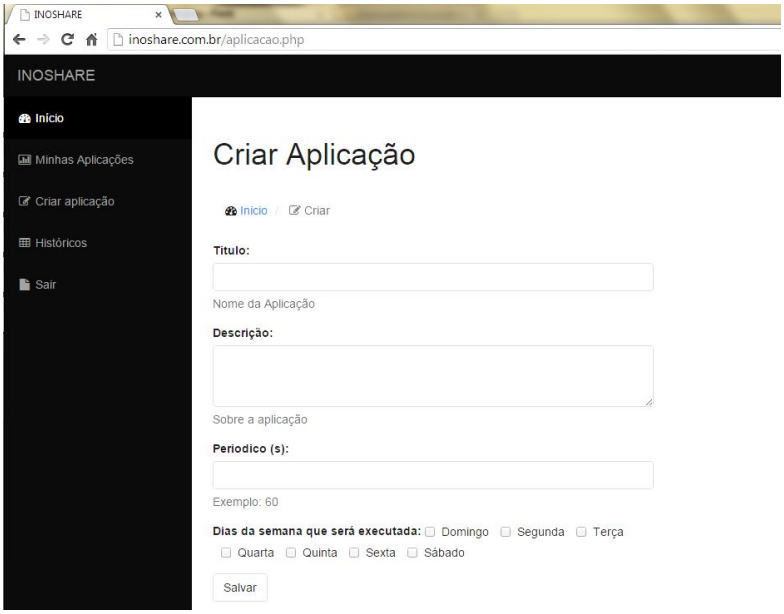
Figura 21 - Visão geral da página "Minhas Aplicações"



Fonte: elaborada pelo autor.

Na figura 22 é possível observar com detalhes a página de cadastro de aplicações. Nessa página o usuário poderá informar os dados primários como, o título da aplicação, período em segundos de espera para ser executada e os dias da semana que deverá ser executada. Cada usuário poderá criar quantas aplicações forem necessárias para suprir a sua necessidade. Uma aplicação sem condições cadastradas não gerará *logs* de execuções.

Figura 22 - Janela de cadastro de aplicações.



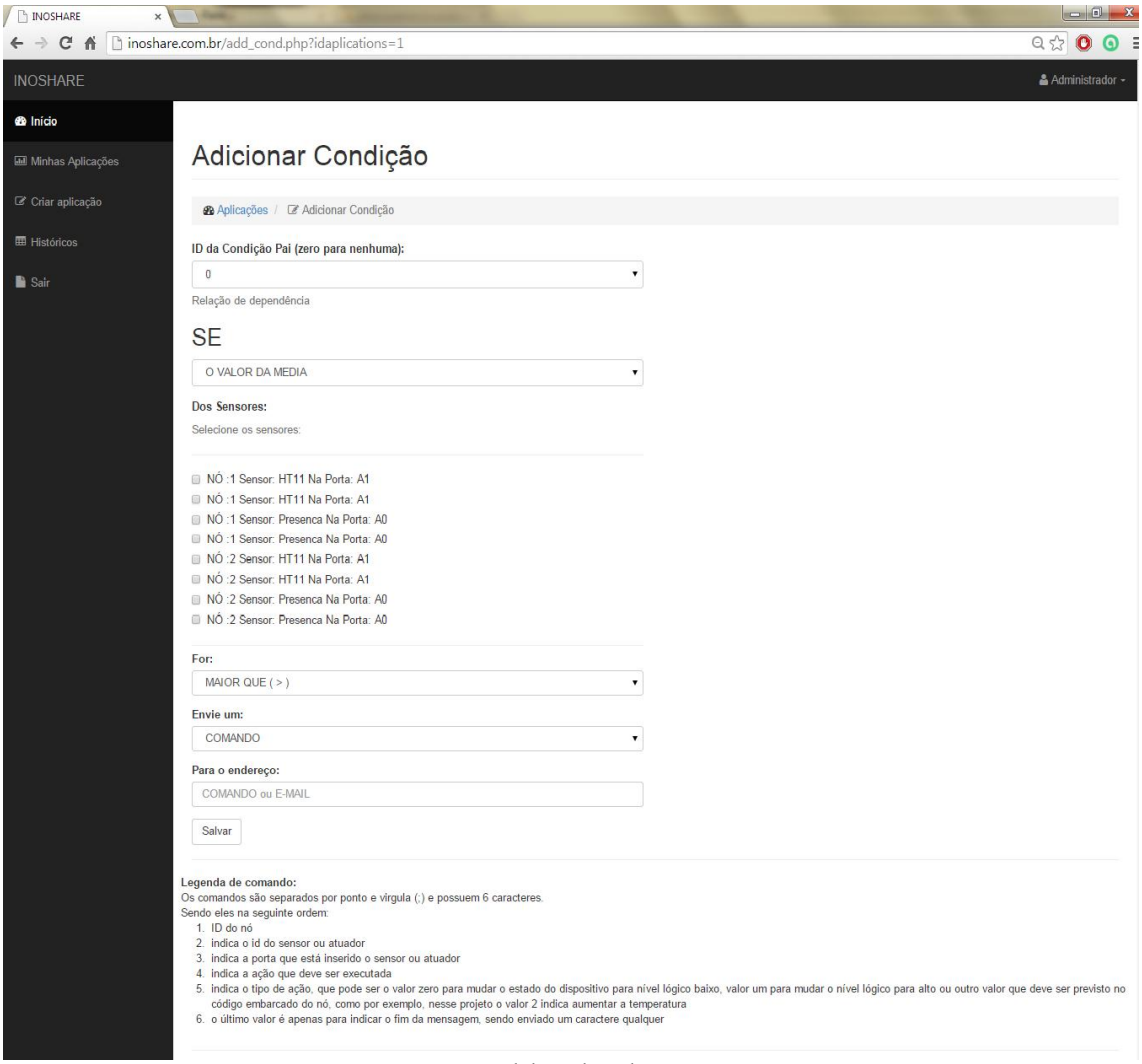
Fonte: elaborada pelo autor.

Uma vez que o usuário cadastra suas aplicações ele poderá a partir de então inserir as condições e ações para cada aplicação. Para que o usuário possa cadastrar uma condição ele deve navegar até a página "Minhas Aplicações" em seguida clicar no botão "condições" na linha da aplicação e em seguida, dentro da página de condições, clicar no link "Inserir Nova Condição". Na figura 23 podemos ver a estrutura da página de cadastro de condições do INOSHARE. Para que a condição cadastrada funcione, o usuário deverá informar pelo menos um endereço de sensor e o comando que deverá ser executado em caso da condição ser verdadeira.

Na página de condições podemos visualizar a lista de condições cadastradas para a aplicação e suas características como, o identificador da condição de dependência (caso exista), a condição propriamente dita, o tipo de comando a ser enviado e o comando a ser enviado caso a condição seja verdadeira. Cada condição possui a opção de visualizar os gráficos de execuções.

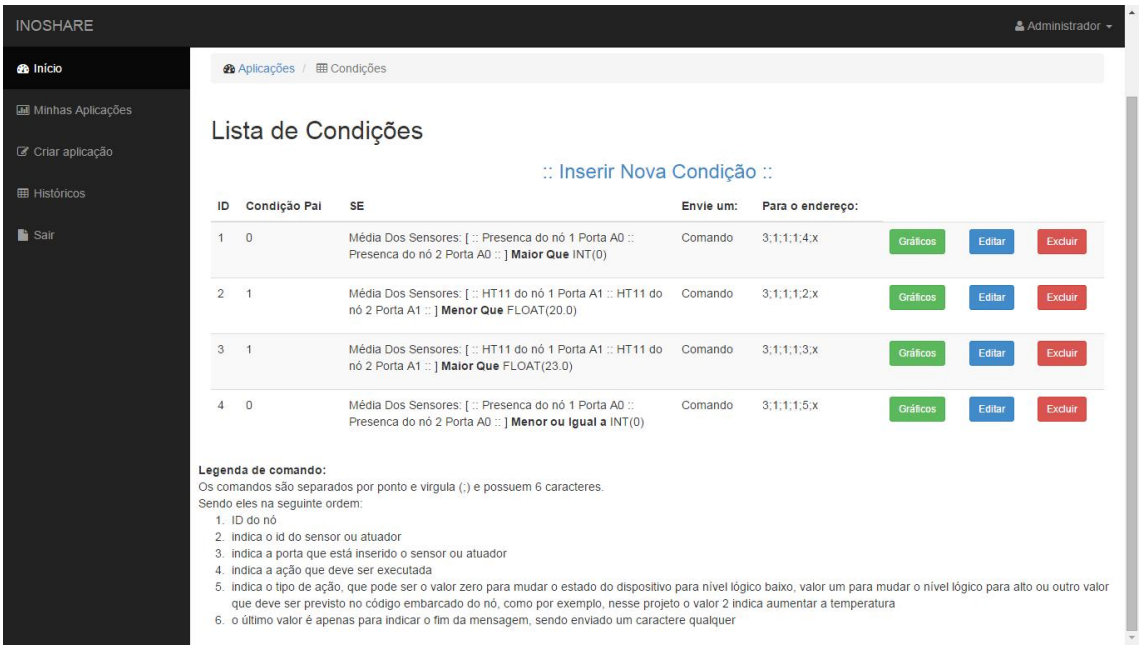
O gráfico disponibilizado para o usuário em cada condição informa os momentos em que a aplicação foi executada, o valor a ser comparado e o valor dos dados coletados pela RSAC. Na janela de gráfico o usuário ainda pode realizar um filtro informando a data e hora inicial e a quantidade de registros que o usuário quer visualizar no gráfico. Na figura 30 podemos observar um exemplo de filtragem. Na imagem, a data inicial selecionada foi 26/09/2014 às 14 horas zero minutos e zero segundos, e a quantidade de registros igual a 100.

Figura 23 - Página da cadastro de condição e ação.



Fonte: elaborada pelo autor.

Figura 24 - Página de listagem de condições de aplicação.



Fonte: elaborada pelo autor.

Figura 25 - Exemplo de gráfico de execuções de uma condição.

Fonte: elaborada pelo autor.

Figura 26 - Exemplo de filtragem de resultados para exibição no gráfico de execuções de uma condição.

Fonte: elaborada pelo autor.

A página "Históricos" lista para o usuário os últimos 100 registros de execuções das aplicações. Uma observação importante é que uma aplicação pode ser registrada como executada mesmo que nenhuma de suas condições sejam verdadeiras, pois mesmo que não tenha ocorrido a execução de comandos, as condições podem ter sido testadas pelo INOSHARE *Core*. Na figura 27 podemos observar como o usuário visualiza o histórico de execução de aplicações.

Figura 27 - Página de histórico de execuções das aplicações.

Fonte: elaborada pelo autor.

6 Experimentos e Resultados

Este capítulo descreve a metodologia dos experimentos realizados para avaliar os resultados obtidos no funcionamento do INOSHARE e nas aplicações cadastradas. Com esse intuito, foram cadastradas duas aplicações: (i) uma replica da aplicação original ClimaDuino (CRUZ, 2013) e (ii) uma aplicação de detecção de risco de fogo.

Essa seção está organizada da seguinte forma: na seção 6.1 é descrito o cenário de testes e suas características e seção 6.2 são apresentadas as métricas utilizadas para a avaliação dos resultados.

6.1 Descrição do cenário dos testes

Durante o período de testes, os sensores de presença utilizados originalmente na aplicação ClimaDuino, apresentaram instabilidade, sendo necessária a compra de novos sensores do mesmo modelo para sanar o problema. Outros problemas como instabilidade na rede e problemas no funcionamento da nuvem do campus, forçaram a aquisição de serviços externos ao campus como o da Amazon Web Services (AWS, 2014).

Foram criadas e testadas duas aplicações (seção 7.3) e todos os serviços executados na nuvem foram instalados em uma instância da *Amazon Web Services* com medidas de recursos de hardware especificados na seção 5.4. As aplicações foram executadas durante um período de três dias (de 24 a 26 de setembro de 2014) nos horários de 07:45 às 21:20 em cada um dos dias de teste. O período de testes durou no total 40 horas e 45 minutos.

6.2 Métricas

Com o objetivo de analisar o consumo de recursos de processamento e memória pelo sistema, a instância da nuvem que executou os serviços utilizados no INOSHARE foi monitorada e seus índices de uso de CPU e memória foram registrados. Durante todo o período de testes, os valores de consumo de CPU e memória foram coletados a cada 10 segundos.

Para analisar a integração da RSSF com a nuvem, os comandos enviados para a RSAC foram sumarizados assim como os comandos que foram executados pela rede e confirmada a execução. Além disso, para calcular a duração média em que os dados coletados pela RSSF são salvos no banco de dados *mogodb*, foi monitorado o instante em que os dados chegaram

ao *gateway* e em seguida subtraído do instante que foram inseridos no banco de dados. O tempo médio de confirmação de execução dos comandos recebidos da nuvem pelo *gateway* e enviados para a RSSF também oi medido em todo o período de testes.

Para analisar os dados coletados de temperatura, umidade relativa do ar e presença, foram calculadas as médias desses dados minuto a minuto. Para os valores de temperatura e umidade relativa do ar foram calculadas também as médias diárias, desvio padrão e variâncias dos valores coletados.

7 Análise dos resultados

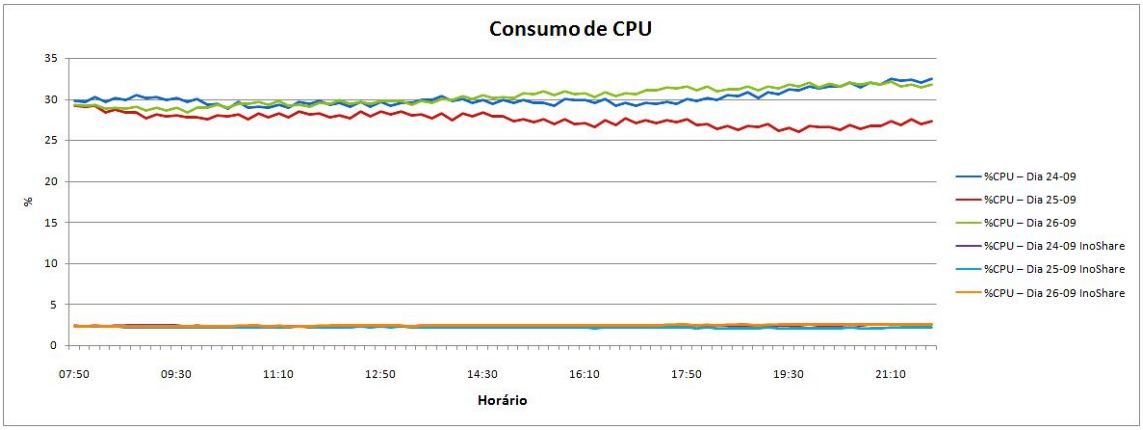
7.1 Análise do sistema: Memória e CPU

O objetivo dessa análise é identificar a quantidade de recursos de CPU e memória necessários para executar o estudo de caso. A tabela 3 mostra as porcentagens médias de consumo de CPU da instância pelos serviços inseridos na nuvem e o consumo individual do processo inoshare (o qual representa o INOSHARE *Core)*. O processo inoshare representa 7,9% do consumo geral dos recursos de CPU. Conforme apresentado na subseção 6.4, proporcionalmente o consumo de CPU do processo inoshare foi de em média 59,35MHz. Já o consumo geral foi de em média 750,90MHz. A Figura 28 apresenta dados de consumo de CPU em médias de 10 minutos para uma melhor visualização.

Tabela 3 - Médias de porcentagem de consumo de CPU de toda a aplicação e do processo inoshare.

	Todos os processos	***Processo inoshare***
Dia 24/09	30,10%	2,37%
Dia 25/09	27,55%	2,17%
Dia 26/09	30,34%	2,39%
Variância das médias	2,39%	0,014%
Média Geral	**29,33%**	**2,31%**

Figura 28 - Gráfico de consumo de CPU.



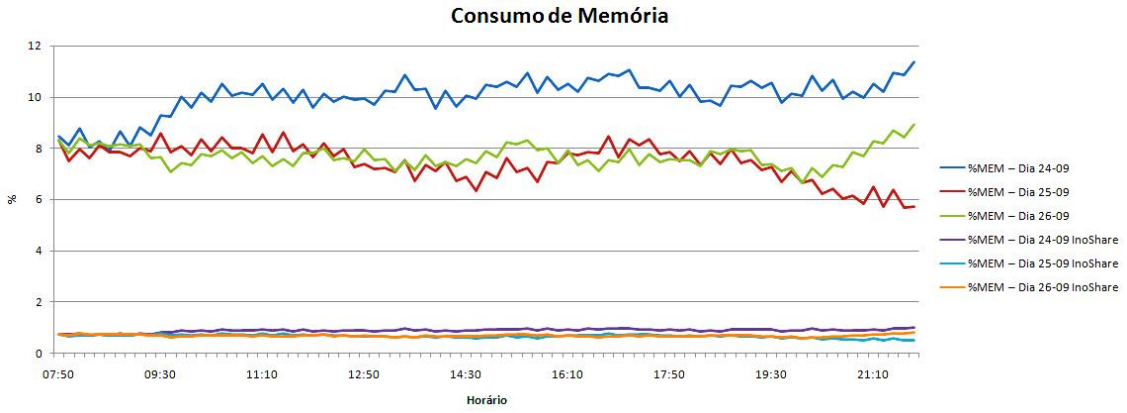
Fonte: elaborada pelo autor.

A tabela 4 nos mostra a porcentagem de consumo médio de memória RAM da instância por todos os processos e o consumo de memória do processo inoshare. Durante todo o período de testes, o consumo de memória pelo processo inoshare foi em média 15,21MB enquanto o consumo geral foi de aproximadamente 171,76MB. Na figura 29 podemos observar as médias de porcentagem de consumo de memória RAM em períodos de 10 minutos.

Tabela 4 - Médias de porcentagem de consumo de memória RAM de toda a aplicação e do processo inoshare.

	Todos os processos	***Processo inoshare***
Dia 24/09	10,03%	0,88%
Dia 25/09	7,43%	0,65%
Dia 26/09	7,69%	0,68%
Variância das médias	2,047%	0,016%
Média geral	**8,38%**	**0,74%**

Figura 29 - Gráfico de consumo de memória RAM.



Fonte: elaborada pelo autor.

Os picos de uso de CPU e memória foram de 32,47% e 11,37% respectivamente, o que representa 831,23MHz de uso de CPU e 232,85MB de memória RAM.

7.2 Análise da integração da RSSF com a nuvem

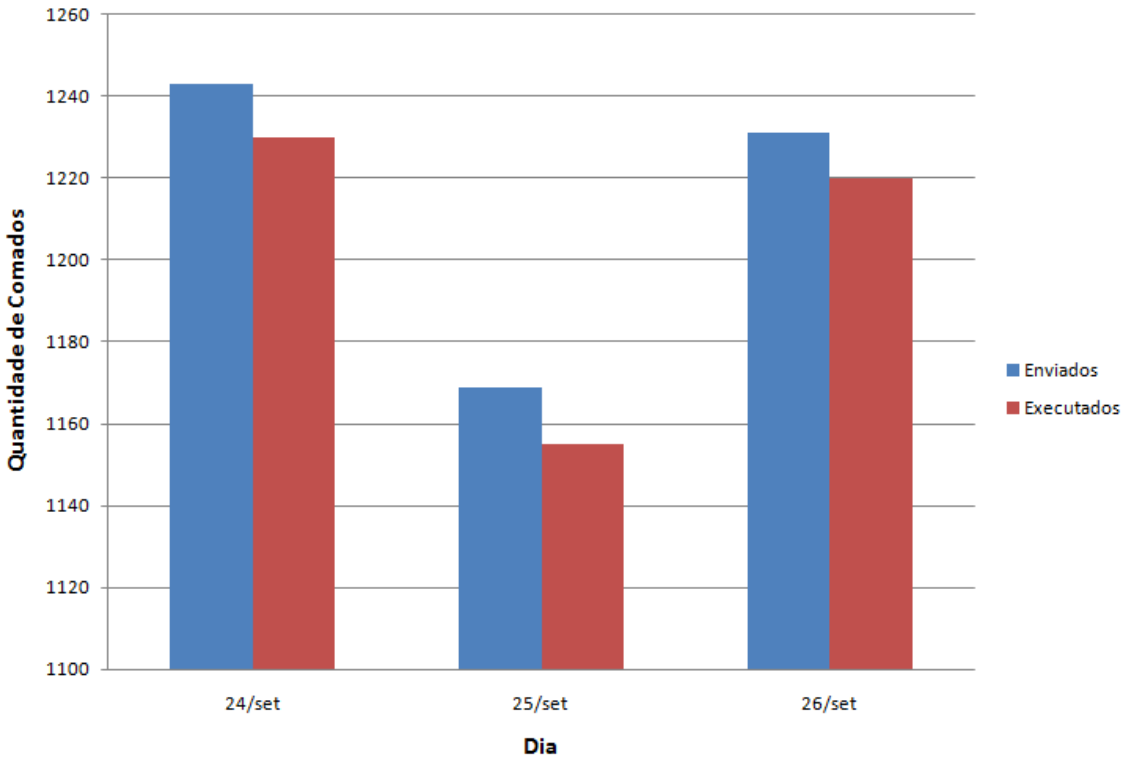
O objetivo dessa análise é identificar o tempo de resposta da nuvem para o armazenamento dos dados coletados pela RSAC, o tempo de resposta da rede na execução de comandos. Com essa análise, espera-se que ao implementar o INOSHARE tenha-se uma visão exata do tempo necessário para que os dados que serão utilizados pelas aplicações estejam salvos no banco de dados.

Na tabela 5 é possível observar o número total de comandos não executados e o aproveitamento total de comandos enviados e executados a cada dia de testes. A média geral de aproveitamento dos comandos enviados para a rede foi de 98,95%. Obtivemos uma média geral de 12 comandos que não foram executados e ou não tiveram suas respectivas execuções confirmadas. Na figura 30 é possível visualizar a quantidade total de comandos enviados da nuvem para a RSAC, além da quantidade total de comandos executados em cada dia de testes.

Tabela 5 - sumarização de comandos não enviados e porcentagem de aproveitamento.

Dia	Comandos perdidos e ou não confirmados	Aproveitamento
24/09	13	98,95%
25/09	14	98,80%
26/09	17	99,10%

Figura 30 - Total de comandos enviados e executados.

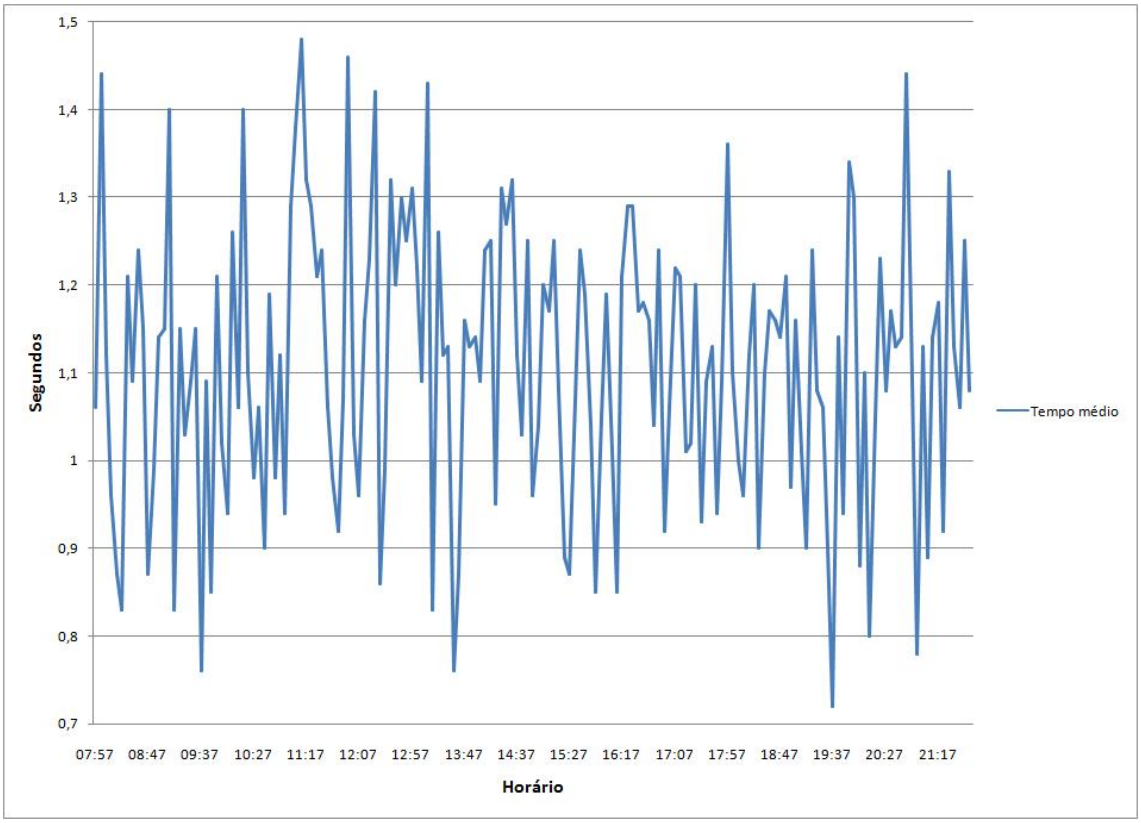


Fonte: elaborada pelo autor.

O resultado do cálculo da duração média para que os dados coletados pela RSAC sejam salvos no banco de dados *mogodb* apontou uma média geral de 1,10 segundos de duração para que a operação fosse concluída. O tempo máximo de duração para que os dados fossem inseridos no banco de dados foi de 1,48 segundos enquanto que o tempo mínimo foi de 720 milissegundos. A variância do tempo médio de duração foi de 25 milissegundos. Na figura 31 temos um gráfico apresentando a média de duração para inserir os dados coletados pela RSSF na nuvem. Os valores do gráfico apresentam a média dos três dias de testes.

Com essa análise de tempo de resposta para a inserção de dados no banco, podemos perceber que, nesse estudo de caso, não seria possível ter com exatidão a média de dados coletados pela RSAC se os mesmos fossem coletados a cada segundo.

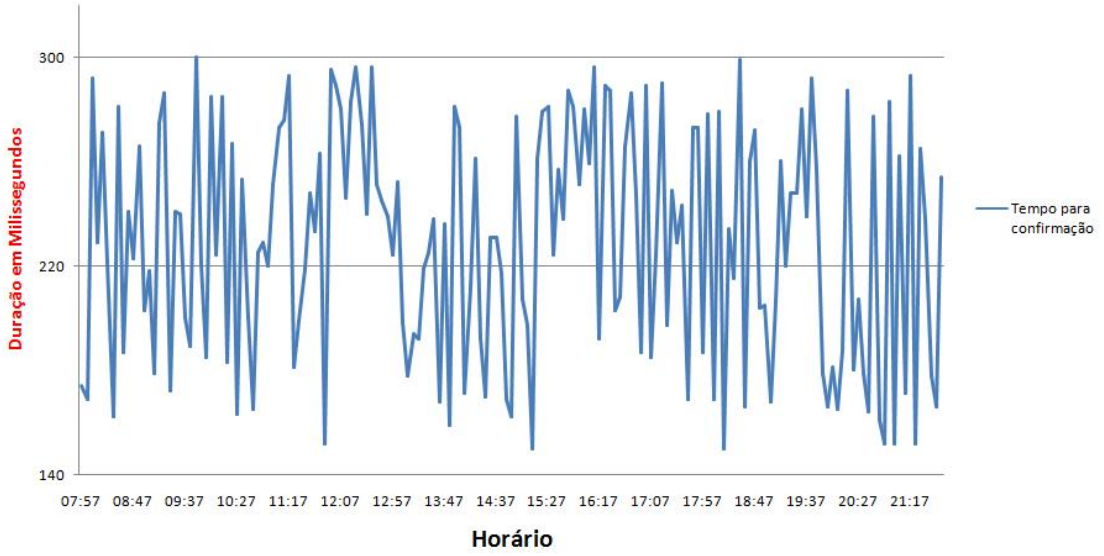
Figura 31 - tempo médio de duração para inserir os dados coletados pela RSSF na nuvem.



Fonte: elaborada pelo autor.

O tempo médio de confirmação de execução dos comandos recebidos da nuvem pelo *gateway* e enviados para a RSAC obteve uma média de 218 milissegundos de duração. O período mais curto de duração para confirmação de execução foi de 150 milissegundos e o mais longo foi de 300 milissegundos. Na figura 32 o gráfico mostra as médias de duração de confirmação de comandos dos três dias em períodos de 5 segundos. Com a análise do tempo de resposta da RSAC para executar os comandos, podemos observar que aplicações sensíveis ao tempo de resposta ou a variação do tempo de resposta, não se aplicariam ao INOSHARE utilizando uma rede no perfil utilizado nesse estudo de caso.

Figura 32 - gráfico de tempo médio de duração para confirmação de execução de comandos.



Fonte: elaborado pelo autor.

Os dados coletados pela RSSF que foram salvos no banco de dados *mongodb* somaram ao todo 289.692 registros coletados segundo a segundo que somou um total de 52,51MB de armazenamento consumidos.

7.3 Análise das aplicações

Inicialmente para que o funcionamento do INOSHARE pudesse ser avaliado tornou-se necessário recriar a aplicação originalmente proposta na aplicação ClimaDuino. Segundo CRUZ (2013) o sistema de controle de centrais de ar condicionado possui as seguintes regras:

- Se houver presença a central de ar condicionado deve ser ligada;
- Se não houver presença a central de ar condicionado deve ser desligada;
- Se houver presença e a temperatura local for maior que 23º C, o atuador deve enviar um comando para baixar a temperatura; e,
- Se houver presença e a temperatura local for menor que 20º C, o atuador deve enviar um comando para aumentar temperatura.

No ClimaDuino os cálculos de temperatura são medidos com base na média de temperatura coletados pelos sensores e o cálculo é realizado uma vez por minuto. Para recriar a aplicação utilizando o INOSHARE, primeiramente foi cadastrado um usuário no sistema e

em seguida cadastrada a aplicação denominada "Sistema de Controle de ar Condicionado". A periodicidade da aplicação foi definida em 60 segundos. Quatro condições foram cadastradas para a aplicação sendo elas:

1. Se a média dos sensores de presença for maior que zero, então um comando de ligar a central de ar condicionado deve ser enviado.
2. Se a condição 1 for verdadeira e a média dos sensores de temperatura for maior que 23.00, então um comando para baixar a temperatura deve ser enviado;
3. Se a condição 1 for verdadeira e a média dos sensores de temperatura for menor que 20.00, então um comando de aumentar a temperatura deverá ser enviado.
4. Se a média dos sensores de presença for menor ou igual a zero, então um comando para desligar a central de ar condicionado deverá ser enviado.

A segunda aplicação cadastrada no INOSHARE foi um sistema de detecção de fogo. Segundo a FUNCEME (2014), um dado de temperatura e umidade são relevantes quando a temperatura está acima de 30º C e a umidade relativa do ar está abaixo de 40%. Baseado nessas informações, uma aplicação denominada "Risco de Fogo" foi cadastrada no banco de dados com periodicidade de 60 segundos. Duas condições foram criadas, sendo elas:

1. Se a média dos sensores de temperatura for maior que 30.00, não faz nada.
2. Se a condição 1 for verdadeira e a média dos sensores de umidade relativa do ar for menor que 40.00, então envie um e-mail informando a existência do risco de fogo.

a) Aplicação de controle de ar condicionado

O resultado esperado para a aplicação de controle de centrais de ar condicionado é inicialmente que a temperatura do ambiente controlado se mantenha sempre dentro da média de conforto, que é entre 20º C e 23º C. Para que isso seja possível é necessário coletar dados de temperatura do ambiente e calcular médias (CRUZ, 2013). Na tabela 6 temos os valores de média diária de temperatura, desvio padrão e variância da temperatura coletada dentro do ambiente controlado pela aplicação de gestão de estação de ar condicionado executada no INOSHARE.

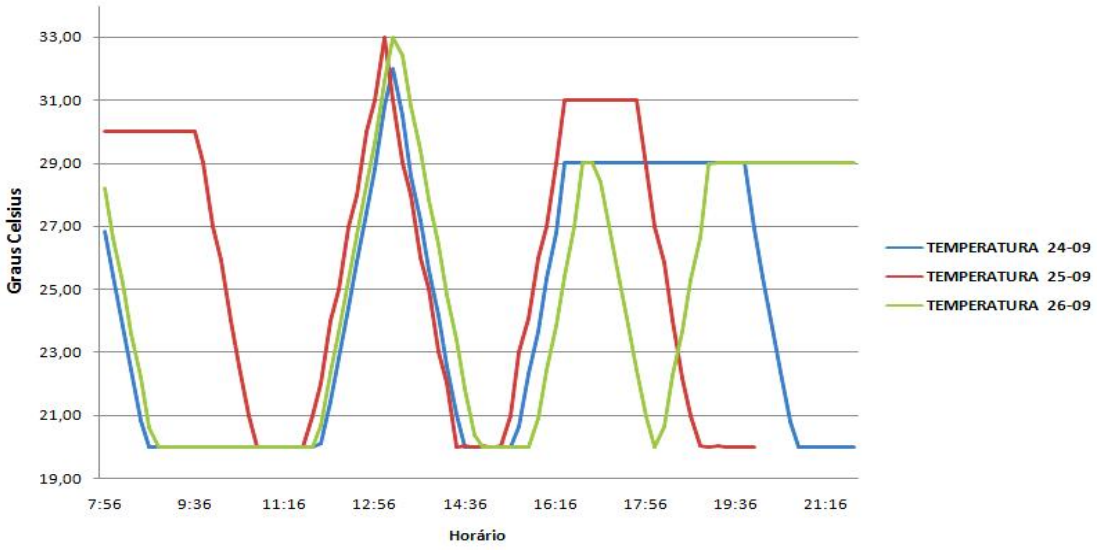
Tabela 6 - dados de variância, média de temperatura e desvio padrão dos dados de temperatura coletados pela RSSF.

Dia	Variância	Média da temperatura diária	Desvio padrão
24/09	16,46º C	24,04º C	4,06º C
25/09	19,77º C	25,51º C	4,45º C
26/09	16, 58º C	24, 55º C	4,07º C

O modelo de sensor de presença e movimento utilizado na rede, quando detecta presença ou movimento, em seu pino de saída libera uma tensão entre 1,24 volts e 2,55 volts aleatórios. Quando não detectam nenhuma presença retornam zero (ver anexo A). No Arduino, qualquer voltagem inserida em uma porta analógica retorna o valor 1. Nesse projeto o pino de saída do sensor de presença foi inserido em uma porta analógica. Portas analógicas no Arduino podem receber até 5 volts. Quando recebem 5 volts retornam o valor 1024, e esse valor de saída é proporcional ao valor de entrada, logo se inserirmos o valor de zero volts a saída também será zero (ARDUINO S.A., 2014).

Na figura 36 podemos visualizar o gráfico de medição de temperatura nos dias de testes dentro de um laboratório no campus Quixadá. Na figura 33 temos o gráfico de presença e movimento relativos ao mesmo período do de temperatura. O que podemos notar nesses dois gráficos é que a temperatura obteve uma queda em seu valor em todos os períodos em que houveram detecção de presença e movimento. Podemos ainda notar que a temperatura sofreu um aumento significativo quando não houveram detecções de presença no local.

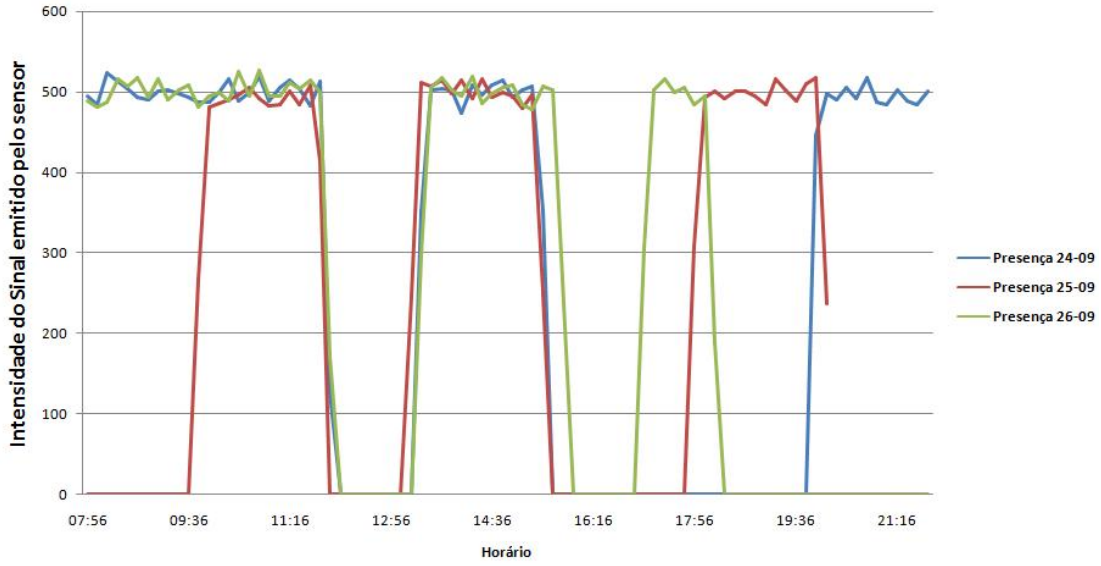
Figura 33 - dados de coleta de temperatura.



Fonte: elaborada pelo autor.

Durante os testes o valor médio de queda de temperatura foi de aproximadamente 0,13° C por minuto, o valor médio de elevação de temperatura do ambiente quando a central de ar condicionado estava desligada ficou entre 0,07° C e 0,10° C por minuto. Alguns fatores externos podem contribuir diretamente na mudança climática do interior do laboratório, como por exemplo, a existência de computadores ligados sem o uso. Esse tipo de prática tende a aumentar a aceleração da elevação da temperatura, o que facilita o rápido equilíbrio térmico do ambiente interno com o ambiente externo (CINDRA; TEIXEIRA, 2004).

Figura 34 - dados de coleta de presença e movimento.



Fonte: elaborada pelo autor.

A tabela 7 apresentam informações sobre o tempo total em que a central de ar condicionado permaneceu em funcionamento, período que permaneceu desligada e o valor economizado em cada dia de testes. Os valores de economia foram gerados com base nos cálculos realizados no ClimaDuino (CRUZ, 2013). Segundo a COELCE (2014) no momento da realização desse projeto a tarifa por *kilowatt* (kW) é de R$ 0,42870, e as fórmulas que correspondem ao preço de um kW por hora e uso (R$/kWh) são a seguintes:

- ***kWh = (Potência do equipamento * N° de horas utilizadas * N° uso por mês) / 1000.***
- ***Total a pagar = kWh * tarifa.***

A potência do equipamento testado é de 4000W segundo o fabricante Gree (GREE, 2014). Nesse contexto podemos afirmar que o valor de economia por dia de testes é igual ao período que as centrais permaneceriam em funcionamento sem a movimentação de pessoas. Segundo CRUZ (2013) é possível que os equipamentos permaneçam em média, entre 15 e 30 minutos em funcionamento desnecessariamente entre os intervalos de aula, horários de refeições e o final das aulas. Em uma análise comparativa em outros laboratórios conseguimos constatar que a média de tempo em que as centrais permanecem em funcionamento desnecessariamente é de 25 minutos por intervalo. Nos dias de testes tivemos respectivamente 2, 3 e 3 intervalos de tempo entre aulas.

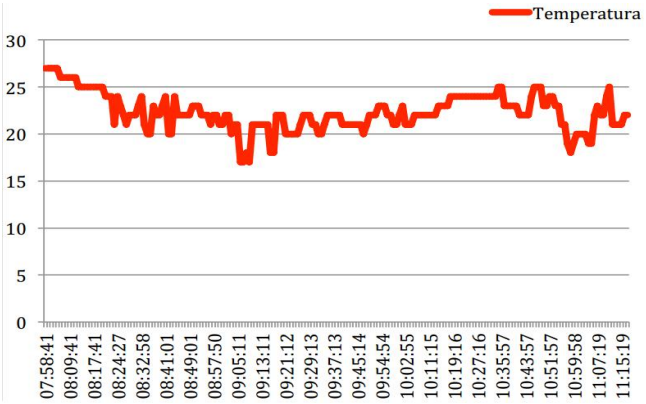
Tabela 7 - dados de funcionamento das centrais de ar condicionado, economia por dia de testes.

Dia	*Funcionamento*	*Desligado*	*Economia*
24/09	510 minutos	340 minutos	R$ 1,4289
25/09	410 minutos	330 minutos	R$ 2,1435
26/09	500 minutos	360 minutos	R$ 2,1435

Por fim obtivemos uma média de economia para a estação de ar condicionado controlada de R$ 1,9053. O que representa uma estimativa de economia mensal de em média R$ 57,15 por central de ar condicionado. No campus de Quixadá existem até o momento somando salas de aula e laboratórios um total de 22 estações de ar condicionado de 4000W cada, o que é equivalente a uma economia de em média R$ 1.257,30 mensais.

O ClimaDuino apenas nos apresenta um trecho de aproximadamente 8 horas da coleta de temperatura do ambiente de testes. Podemos observar na figura 39 que, dentro da amostra, em alguns momentos a temperatura do ambiente esteve abaixo de 20° C, o que podemos concluir que a aplicação em determinados momentos não obteve um funcionamento dentro do esperado não mantendo a temperatura dentro da faixa de 20º C a 23º C. Nesse ponto podemos observar que a aplicação que foi executada no INOSHARE apresentou uma maior confiabilidade no objetivo principal.

Figura 35 - gráfico de coleta de temperatura do ClimaDuino.



Fonte: CRUZ 2013.

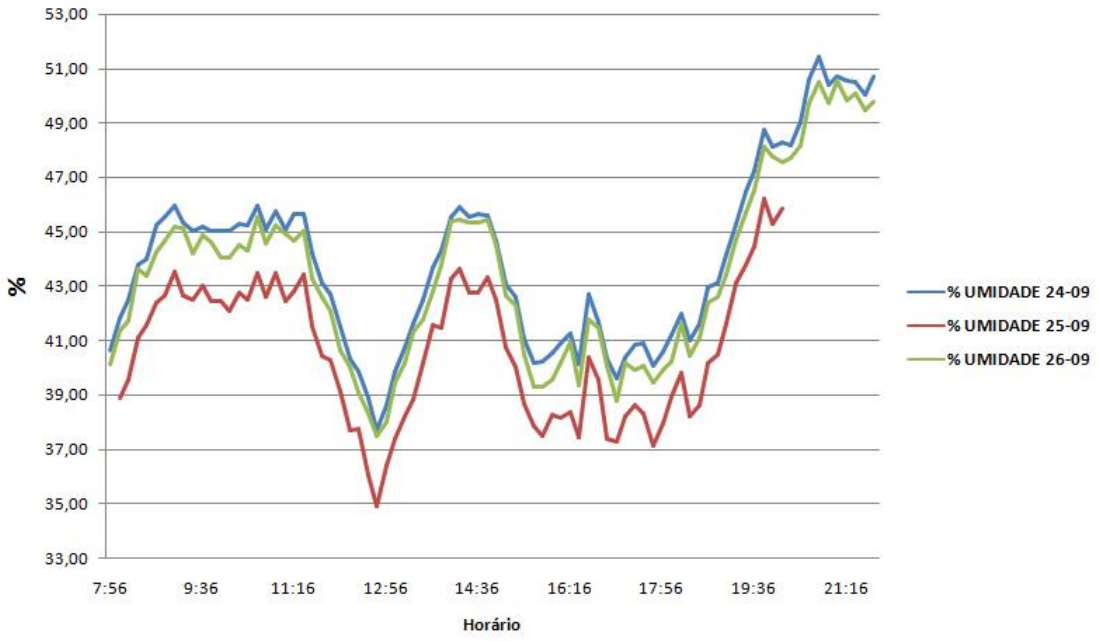
b) Aplicação de detecção de risco de fogo

Nesse trabalho foi considerado que um dado de temperatura é relevante quando está acima de 30° C e a umidade relativa do ar está abaixo de 40% (FUNCEME, 2014). Na tabela 8 podemos visualizar dados relativos a umidade relativa do ar em cada dia de coleta no ambiente de testes. A média geral em todo o período de testes foi de 42,68% e a variância das médias foi de 3,20 pontos percentuais.

tabela 8 - valores de média, variância e desvio padrão da umidade relativa do ar do período de testes.

Dia	**Média**	**Variância**	**Desvio padrão**
24/09	44,00%	11,06	3,32
25/09	40,64%	6,49	2,54
26/09	43,40%	11,05	3,32

Figura 36 - gráfico de coleta de umidade relativa do ar

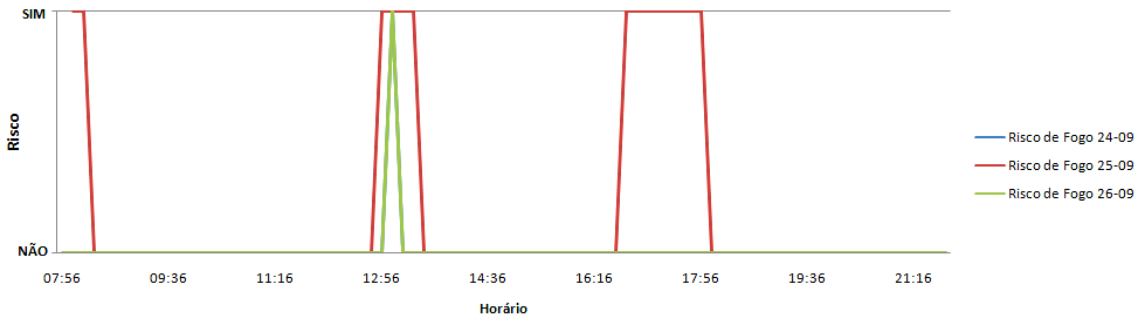


Fonte: elaborado pelo autor.

No gráfico de coleta da umidade relativa do ar (figura 36) podemos observar que o comportamento padrão da umidade é se manter constante pela manhã, diminuir rapidamente no período da tarde e aumentar rapidamente no período da noite.

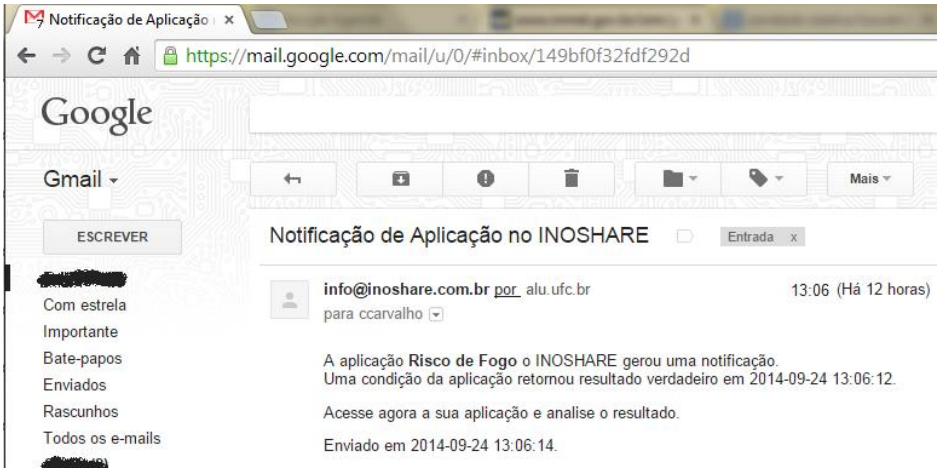
No primeiro dia do período de testes obtivemos a menor média de umidade relativa do ar em comparação com o restante do período. A aplicação de risco de fogo deveria informar ao usuário via e-mail sempre que a média dos dados coletados pelos sensores de temperatura fosse maior que 30° C e a média dos dados coletados pelos sensores de umidade fosse menor que 40%. No total obtivemos 15 mensagens enviadas pelo INOSHARE sendo que 14 dessas foram enviadas no segundo dia de testes. A figura 37 nos mostra os instantes onde a aplicação identificou a possibilidade de risco de fogo conforme as condições nela cadastradas.

Figura 37 - gráfico de risco de fogo.



Fonte: elaborado pelo autor.

Figura 38 - exemplo de e-mail de alerta sobre a aplicação de risco de fogo.



Fonte: elaborado pelo autor.

Durante todo o período de testes o segundo dia obteve 93% das informações que foram compatíveis com os dados de risco de fogo inseridos no INOSHARE. Dos períodos que foram determinantes para identificar risco de fogo durante o período de testes, 53% ocorreram entre as 16:46 e 18:00 horas do segundo dia de testes.

Segundo dados do Instituto Nacional de Meteriologia (INMET) e a Fundação Cearense de Meteriologia e Recursos Hídricos FUNCEME, no segundo semestre do ano de 2014 a média de umidade relativa do ar de cidades do sertão central pode atingir valores entre 31% e 40% (INMET, 2014; FUNCEME, 2014), o que representa um alto índice de risco de incêndios florestais.

8 CONSIDERAÇÕES FINAIS

Esse trabalho propôs o INOSHARE, uma ferramenta para proporcionar o compartilhamento dos recursos de uma RSAC, criar aplicações e gerenciá-las. Nesse trabalho ainda foi proposto a implementação da integração da RSAC com a computação em nuvem, executando as suas aplicações em ambiente remoto. Aqui serão descritas as conclusões obtidas a partir dos resultados apresentados no capítulo anterior.

8.1 Conclusões

O INOSHARE pode contribuir para o compartilhamento de recursos por múltiplas aplicações em RSAC's. Utilizando a computação em nuvem para armazenar e executar as aplicações, foi observado um baixo consumo de recursos computacionais. Esse aspecto permite criar múltiplas aplicações de forma escalável, dentro dos recursos existentes na rede de sensores. Nesse trabalho, foi apresentado um estudo de caso contendo inicialmente uma RSAC que não compartilhava seus recursos, que com pequenas modificações, tornaram possível o compartilhamento de recursos por múltiplas aplicações.

O INOSHARE permite a execução das aplicações criadas e a supervisão das execuções pelo usuário por meio de uma interface web. Os dados coletados ainda podem ser utilizados em aplicações fora do INOSHARE através do acesso via *Web Service*. No INOSHARE foi realizada a integração da RSAC com a computação em nuvem apresentado resultados favoráveis a essa implementação. O uso da computação em nuvem proporciona vantagens como a escalabilidade de armazenamento e alta disponibilidade.

Na avaliação foi possível notar que as aplicações não interferiram entre si, executando de forma paralela como esperado. A aplicação de controle de ar condicionado foi capaz de manipular a temperatura do ambiente enquanto a aplicação de detecção de fogo pode interpretar as condições criadas e executar os comandos que foram determinados.

Considera-se limitações nessa proposta: (i) o tamanho da rede utilizada no estudo de caso (apenas 4 nós) e (ii) a quantidade de aplicações dadas as limitações de recursos existentes na rede. O número de aplicações e as condições criadas para testá-las não foram suficientes para avaliar todos os recursos criados, como por exemplo, aplicações que executem comandos no mesmo atuador e que possuam grandes chances de enviar comandos no mesmo momento. No entanto, o ambiente de testes foi real, onde foi possível observar as variações de tempo de resposta da rede e da nuvem e avaliar o comportamento da rede.

8.2 Trabalhos Futuros

Pretende-se testar o INOSHARE em uma rede com uma quantidades maiores de sensores e atuadores, cujo tempo de funcionamento das aplicações seja capaz de coletar dados em grandes escalas. Além disso, pretende-se testar o INOSHARE com a execução de um número maior e variável de aplicações. Assim, permite-se analisar com mais precisão a

escalabilidade do armazenamento de dados, o uso de recursos computacionais e o tempo de resposta da rede e nuvem frente a um grande número de comandos e aplicações.

REFERÊNCIAS

GROUP, The Php. **Manual do PHP**. 2014. Disponível em: <http://php.net/manual/pt_BR/index.php>. Acesso em: 13 nov. 2014.

FOUNDATION, Python Software. **Python 2.7.8 documentation**. 2014. Disponível em: <https://docs.python.org/2/>. Acesso em: 13 nov. 2014.

HAN, J.; HAIHONG, E.; LE, G.; DU, J. Survey on NoSQL database. **Pervasive computing and …**, p. 363–366, 2011. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6106531>. Acesso em: 12/11/2014.

MONGODB.INC. **The MongoDB 2.6 Manual**. 2014. Disponível em: <http://docs.mongodb.org/manual/>. Acesso em: 12 nov. 2014.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. **Computer Networks**, v. 52, n. 12, p. 2292–2330, 2008. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128608001254>. Acesso em: 19/3/2014.

YE, F.; PAN, R. A survey of addressing algorithms for wireless sensor networks. **… Networking and Mobile Computing, 2009. WiCom …**, , n. 2007, 2009. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5304082>. Acesso em: 12/11/2014.

ROCHA, A. R.; PIRMEZ, L.; DELICATO, F. C.; et al. **WSNs clustering based on semantic neighborhood relationships. Computer Networks**, v. 56, n. 5, p. 1627–1645, 2012. Elsevier B.V.

HERMETO, R. T.; KRIDI, D. S.; ROCHA, A. R. DA; GOMES, D. G. **A distributed algorithm for semantic collectors election in wireless sensors networks.** Journal of Applied Computing Research, v. 3, n. 1, p. 1–10, 2014.

BARI, N.; MANI, G.; BERKOVICH, S. Internet of Things as a Methodological Concept. 2013 Fourth International Conference on Computing for Geospatial Research and Application, p. 48–55, 2013. IEEE.

AHMED, K., & GREGORY, M. (2011). **Integrating Wireless Sensor Networks with Cloud Computing.** 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks , 364–366.

ALAMRI, A. *et al*. (2013). **A survey on sensor-cloud: architecture, applications, and approaches.** International Journal of Distributed Sensor Networks, Saudi Arabia, Volume 2013 , 18.

BHATTACHARYA, S., SAIFULLAH, A., LU, C., & ROMAN, G.-C. (2010). **Multi-Application Deployment in Shared Sensor Networks Based on Quality of Monitoring.** 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium , 259–268.

DIALLO, O., RODRIGUES, J. J., & SENE, M. (2012). **Real-time data management on wireless sensor networks: A survey.** Journal of Network and Computer Applications, v. 35, n. 3 , 1013–1021.

EFSTRATIOU, C., LEONTIADIS, I., MASCOLO, C., & CROWCROFT, J. (s.d.). **Demo Abstract : A Shared Sensor Network Infrastructure.** 367–368.

FARIAS, C. M., PIRMEZ, L., DELICATO, F. C., & al., e. (2013). **A scheduling algorithm for shared sensor and actuator networks.** The International Conference on Information Networking 2013 (ICOIN) , 648–653.

LEONTIADIS, I., & EFSTRATIOU, C. (2012). **SenShare: transforming sensor networks into multi-application sensing infrastructures.** IEEE Conferece of Wireless Sensor Networks .

MECATRÔNICA ATUAL**. Redes Industriais para sensores e atuadores.** (29 de 07 de 2013). Acesso em 27 de 04 de 2014, disponível em mecatrônica atual: http://www.mecatronicaatual.com.br/educacao/1868-redes-industriais-para-sensores-e-atuadores

ARDUINO S.A. **Arduino.** [site corporativo]. Disponível em: < http://arduino.cc/ >. Acesso em: 27 de Abril de 2014.

SHAH, S., KHAN, F., ALI, W., & KHAN, J. (2013). **A new framework to integrate wireless sensor networks with cloud computing.** IEEE Conferecence , 1–6.

WANG, G., & TANG, J. (2012). **The NoSQL Principles and Basic Application of CaRSACdra Model.** 2012 International Conference on Computer Science and Service System , 1332–1335.

YICK, J., MUKHERJEE, B., & GHOSAL, D. (2008). **Wireless sensor network survey.** Computer Networks, v. 52, n. 12 , 2292–2330.

McROBERTS, M. **Arduino básico.** São Paulo: Novatec, 2011.

MELLIS, D. A. **Arduino - Introduction.** Disponível em: <http://www.arduino.cc/en/Guide/Introduction>. Acesso em: 27 de Abril de 2014.

INMET. Instituto Nacional de Meteriologia. Disponível em: < http://www.inmet.gov.br/sim/gera_graficos.php>. Acesso em: 19 de Novembro de 2014.

ESCHNER, R. H. **Sistema de automação residencial baseado em sensores ZigBee.** Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, 2011, Porto Alegre.

CRUZ, M. M. **CLIMADUINO: Sistema Embarcado de Climatização para Edifícios Inteligentes.** Trabalho de Conclusão de Curso (Tecnólogo em Redes de Computadores) – Universidade Federal do Ceará, 2013, Quixadá.

BRITO, A. V. F.. **Uma Metodologia para a Integração de RSSF e Computação em Nuvem.** Trabalho de Conclusão de Curso (Sistemas de Informação) – Universidade Federal do Ceará, 2014, Quixadá.

DIGI INTERNATIONAL. **XBee.** [site corporativo]. Disponível em: <http://www.digi.com/xbee/>. Acesso em: 27 de Abril de 2014.

GREE. Gree. [site corporativo]. Disponível em < http://www.gree.com.br/>. Acesso em: 18 de novembro de 2014.

CUNHA, T. F. **Controle centralizado de equipamentos de ar condicionado via rede sem fio ZigBee.** Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Instituto Federal de Santa Catarina, 2013, São José.

MICROSTRAINS'S. **What is Sensor-Cloud?.** [site corporativo]. Disponível em: <http://www.sensorcloud.com/system-overview>. Acesso em 29 de Abril de 2014.

SALEIRO, M.; EY, E. **ZigBee: uma abordagem prática.** Disponível em: <http://lusorobotica.com/ficheiros/Introducao_ao_Zigbee_-_por_msaleiro.pdf > Acesso em: 27 de Abril de 2014.

AROCKIAM, L. **Cloud computing: a survey.** … of Internet Computing (…, , n. 2, p. 26–33, 2011. Disponível em: <http://interscience.in/IJIC_Vol1Iss2/paper5.pdf>. Acesso em: 29/4/2014.

SOUSA, F. R. C.; MOREIRA, L. O.; MACÊDO, J. A. de ; MACHADO, J. C.. **Gerenciamento de dados em Nuvem: conceitos, sistemas e desafios.** *In*: Simpósio Brasileiro de Banco de Dados, 2010, Belo Horizonte.

DWIVEDI, **A. Operating Systems for Tiny Networked Sensors: A Survey.** International Journal of …, v. 1, n. 2, p. 152–157, 2009. Disponível em: <http://academypublisher.com/ijrte/vol01/no02/ijrte0102152157.pdf>. Acesso em: 22/5/2014.

BAGAA, M.; CHALLAL, Y.; KSENTINI, A.; DERHAB, A.; BADACHE, N. Data Aggregation Scheduling Algorithms in Wireless Sensor Networks: Solutions and Challenges. **ieeexplore.ieee.org**, v. 16, n. 3, p. 1339–1368, 2014. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6780904>. Acesso em: 12/11/2014.

CHOLIA, S.; SKINNER, D.; BOVERHOF, J. NEWT: A RESTful service for building High Performance Computing web applications. **… Workshop (GCE), 2010**, 2010. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5676125>. Acesso em: 12/11/2014.

Hat.INC, Red. Red Hat Enterprise Linux. 2014. Disponível em: <https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/>. Acesso em: 12 nov. 2014.

CORPORATION, Oracle. MySQL Documentation: MySQL Reference Manuals. 2014. Disponível em: <http://dev.mysql.com/doc/>. Acesso em: 12 nov. 2014.

RIVERO, Laura C.; DOORN, Jorge H.; FERRAGINE, Viviana E.. Encyclopedia of Database Technologies and Applications. Hershey: Idea Group Inc (igi), 2006. 784 p. Disponível em: <http://books.google.com.br/books?id=pVvnl1_9dbMC&printsec=frontcover&hl=pt-BR#v=onepage&q&f;=false>. Acesso em: 12 nov. 2014.

VOGEL, Lars. What is JSON? - Tutorial: Version 1.2. 2014. Copyright © 2011, 2012, 2013, 2014 Lars Vogel. Disponível em: <http://www.vogella.com/tutorials/JSON/article.html>. Acesso em: 13 nov. 2014.

FUNCEME. Fundação Cearense de Metereologia e Recursos Hídricos. 2014. Disponível em: <http://www.funceme.br>

AWS. Amazon Web Services. 2014. Disponível em: < http://aws.amazon.com/free/>.

XU, G.; PANG, J.; FU, X. A load balancing model based on cloud partitioning for the public cloud. **Tsinghua Science and Technology**, v. 18, n. 1, p. 34–39, 2013. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6449405>.

CINDRA, J.; TEIXEIRA, O. Discussão conceitual para o equilíbrio térmico. **Caderno Brasileiro de Ensino de …**, p. 176–193, 2004. Disponível em: <http://nutes2.nutes.ufrj.br/coordenacao/textosapoio/cindra_teixeira20041.PDF>. Acesso em: 18/11/2014.

VIEIRA, L.; LOUREIRO, A. Redes de Sensores Aquáticas. … Brasileiro de Redes …, 2010. Disponível em: <http://homepages.dcc.ufmg.br/~lfvieira/SBRC10-bookchapter.pdf>. Acesso em: 25/11/2014.

MACEDO, Diego Carvalho Pereira; ARAÚJO, Harilton da Silva; SOUSA, Aldir Silva. **Inteligência computacional aplicada à RSSF: controle de densidades de nós**. Abakós, Belo Horizonte, v. 2, n. 2, p.35-47, 28 maio 2014. Semestral. V. 2, n. 2 (2014).

APÊNDICES

APÊNDICE A – Recebendo comandos via porta serial no Arduino

Para que a plataforma Arduino seja capaz de interpretar comandos via porta serial é necessário que seu código ao receber os dados, os interprete e em seguida execute-os. O código fonte abaixo é uma implementação utilizada nesse projeto para que o arduino seja capaz de interpretar comandos.

```
/**
 *Código fonte parte integrante do INOSHARE
 * Aqui o nó coleta dados dos sensores e envia e através do uso de thread, simultaneamente
* recebe e interpreta comandos
 */
#include <dht.h>
#include <ThreadController.h>
#include <Thread.h>

//valores fixos
#define dht_pin A1
#define infra_pin A0
#define address_node 1

//objeto do sensor de temperatura
dht DHT;
//objeto do controlador de threads
ThreadController cpu;
//declaracao das trheads
Thread sendMsg;
Thread readComand;

int x;
void setup(){
  //determina o intervalo de pausa de cada thread
  //e em seguida a funcao que sera executada pela thread
  sendMsg.setInterval(2000);
  sendMsg.onRun(sendStatus);

  readComand.setInterval(1000);
  readComand.onRun(comandRead);

  Serial.begin(115200);
  delay(1000);

  for(int i = 2; i<=13; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
  //arduino Mega 2560, pula portas 14-21 por serem utilizadas como serial
  for(int i = 22; i<=53; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
```

```
  //adiciona ao controlador de threads as threads criadas
  cpu.add(&sendMsg);
  cpu.add(&readComand);

}

void loop(){
  //apenas executo as threads
  cpu.run();
}

//envio de status dos sensores
void sendStatus(){

  float temperatura = 0.0;
  int presenca = 0;
  //presenca = !digitalRead(infra_pin);
  presenca = digitalRead(infra_pin) * 1000;

  DHT.read11(dht_pin);
  temperatura = DHT.temperature;

  printJSON("A0", 1, presenca);
  printJSON("A1", 2, temperatura);

}

/*
* ler o comando recebido
* o comando deve ter a seguinte estrutura
* no;sensor/atuador;porta;acao;x
*/
void comandRead(){

  int noDestino = 0, sensor = 0, porta = 0, acao = 0, tipoAcao = 0, i=0;
  x = 0;

  String str, msg;
  while(Serial.available()){
    //recebe dados da serial e armazena em uma string
    char c = Serial.read();
    str += c;
  }

  msg = str;
  do{
    x = msg.indexOf(";");
    if(x != -1){

      String fatia = msg.substring(0, x);
      msg = msg.substring(x+1, msg.length());
      int valorFatia = fatia.toInt();
      if(i == 0){
        noDestino = valorFatia;
      }
      else if(i == 1){
        sensor = valorFatia;
      }
      else if(i == 2){
        porta = valorFatia;
```

```
      }
      else if(i == 3){
        acao = valorFatia;
      }
      else if(i == 4){
        tipoAcao = valorFatia;
      }
    }

    ++i;
  }
  while(x >= 0);

  //manda comando
  if(porta != 0 && sensor != 0 && noDestino == address_node){
    executeAction(porta, acao, tipoAcao);
  }

}

//executa o comando
void executeAction(int porta, int acao, int tpAcao){
  if(tpAcao == 0){
    digitalWrite(porta, acao);

  }
  else if(tpAcao == 1){
    if(acao == 1){
      tone(porta, 800);
    }
    else if(acao == 0){
      noTone(porta);
    }
  }

}

void printJSON(char porta[2], int sensor, float valor){
  int node = (random(10)%2)+1;
  Serial.print("{\"id_no\": \"");
  Serial.print(node);
  Serial.print("\", \"porta\": \"");
  Serial.print(porta);
  Serial.print("\", \"sensor\": \"");
  Serial.print(sensor);
  Serial.print("\", \"valor\": \"");
  Serial.print(valor);
  Serial.print("\"");
  Serial.print("}");
  Serial.println();
}

//funcao especifica o IR
void acoesIR(int acao){
  /*
    Aqui devem ser colocadas as funcionalidades de controle
   de temperatura
   */

}
```

ANEXOS

ANEXO A – Datasheet PIR Sensor (#555-28027) - Parallax Inc.

PARALLAX

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

PIR Sensor (#555-28027)

General Description

The PIR (Passive Infra-Red) Sensor is a pyroelectric device that detects motion by measuring changes in the infrared levels emitted by surrounding objects. This motion can be detected by checking for a high signal on a single I/O pin.

Features

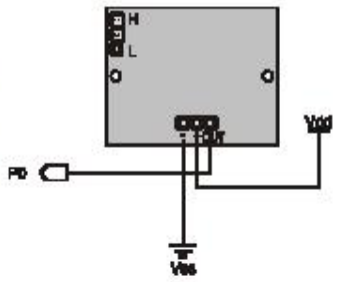
- Single bit output
- Small size makes it easy to conceal
- Compatible with all Parallax microcontrollers
- 3.3V & 5V operation with <100uA current draw

Application Ideas

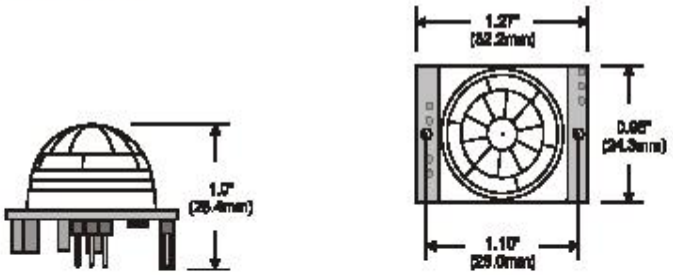
- Alarm Systems
- Halloween Props

Quick Start Circuit

Note: The sensor is active high when the jumper (shown in the upper left) is in either position.



Module Dimensions



Theory of Operation

Pyroelectric devices, such as the PIR sensor, have elements made of a crystalline material that generates an electric charge when exposed to infrared radiation. The changes in the amount of infrared striking the element change the voltages generated, which are measured by an on-board amplifier. The device contains a special filter called a Fresnel lens, which focuses the infrared signals onto the element. As the ambient infrared signals change rapidly, the on-board amplifier trips the output to indicate motion.

Pin Definitions and Ratings

Pin	Name	Function
-	GND	Connects to Ground or Vss
+	V+	Connects to Vdd (3.3V to 5V) @ ~100uA
OUT	Output	Connects to an I/O pin set to INPUT mode (or transistor/MOSFET)

Jumper Setting

Position	Mode	Description
H	Retrigger	Output remains HIGH when sensor is retriggered repeatedly. Output is LOW when idle (not triggered).
L	Normal	Output goes HIGH then LOW when triggered. Continuous motion results in repeated HIGH/LOW pulses. Output is LOW when idle.

Connecting and Testing

Connect the 3-pin header to your circuit so that the minus (-) pin connects to ground or Vss, the plus (+) pin connects to Vdd and the OUT pin connects to your microcontroller's I/O pin. One easy way to do this would be to use a standard servo/LCD extension cable, available separately from Parallax (#805-00002). This cable makes it easy to plug sensor into the servo headers on our Board Of Education or Professional Development Board. If you use the Board Of Education, be sure the servo voltage jumper (located between the 2 servo header blocks) is in the Vdd position, not Vin. If you do not have this jumper on your board you should manually connect to Vdd through the breadboard. You may also plug the sensor directly into the edge of the breadboard and connect the signals from there. Remember the position of the pins when you plug the sensor into the breadboard.

Calibration

The PIR Sensor requires a 'warm-up' time in order to function properly. This is due to the settling time involved in 'learning' its environment. This could be anywhere from 10-60 seconds. During this time there should be as little motion as possible in the sensors field of view.

Sensitivity

The PIR Sensor has a range of approximately 20 feet. This can vary with environmental conditions. The sensor is designed to adjust to slowly changing conditions that would happen normally as the day progresses and the environmental conditions change, but responds by making its output high when sudden changes occur, such as when there is motion.

Resources and Downloads

Check out the PIR Sensor product page for example programs and more:

http://www.parallax.com/detail.asp?product_id=555-28027

Source Code

BASIC Stamp® 1 Program

This program will display the current status of the output pin from the PIR Sensor connected to P0 by lighting an active high LED connected to P1 when motion is detected.

```
' ================================================================================
'   File...... PIR_Simple.bs1
'   Purpose... Show Output State Of PIR Sensor
'   Author.... Parallax, Inc.
'   E-mail.... support@parallax.com
'   Started... 12-14-2005
'   {$STAMP BS1}
'   {$PBASIC 1.0}
'
' -----[ Program Description ]---------------------------------------------

' This program displays the current state of the PIR Sensor connected to P0
' by lighting an active high LED connected to P1 when motion is detected.

' -----[ I/O Definitions ]-------------------------------------------------

SYMBOL          PIR =   PIN0            ' I/O Pin For PIR Sensor
SYMBOL          LED =   PIN1            ' I/O Pin For LED

' -----[ Initialization ]--------------------------------------------------

LET             DIRS =  %00000010       ' Set Pin Directions

' -----[ Program Code ]----------------------------------------------------

Main:
  LET LED = PIR
  GOTO Main
```

BASIC Stamp® 2 Program

This program will display the current status of the output pin from the PIR Sensor connected to P0 using the Debug Terminal.

```
' ================================================================================
'   File...... PIR_Simple.bs2
'   Purpose... Show Output State Of PIR Sensor
'   Author.... Parallax, Inc.
'   E-mail.... support@parallax.com
'   Started... 12-14-2005
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' -----[ Program Description ]---------------------------------------------

' This program displays the current state of the PIR Sensor connected to P0
' on the DEBUG screen.
```

```
' -----[ Program Code ]----------------------------------------------------

Main:
  DO
    DEBUG HOME, BIN1 IN0                ' Display Status Of P0 At Home Pos.
    PAUSE 100                           ' Small Delay
  LOOP                                  ' Repeat Forever
```

SX Microcontroller Application

This program will display the current status of the output pin from the PIR Sensor connected to RC.7 by lighting an active high LED connected to RC.6 when motion is detected.

```
' ===========================================================================
'
'   File..... PIR_Simple.SXB
'   Purpose... Demonstrate Reading The PIR Sensor
'   Author.... Parallax, Inc.
'   E-mail.... support@parallax.com
'   Started... 12-14-2005
'
' Program Description
' ---------------------------------------------------------------------------

' This program will display the status of the output of the PIR sensor
' connected to RC.7 by lighting an active high LED connected to RC.6 when
' motion is detected.  Use a 220 or 330 ohm series resistor with the LED.

' Device Settings
' ---------------------------------------------------------------------------

DEVICE          SX28, OSC4MHZ, TURBO, STACKX, OPTIONX
FREQ            4_000_000

' IO Pins
' ---------------------------------------------------------------------------

PIR             VAR         RC.7            ' I/O Pin For PIR Sensor
LED             VAR         RC.6            ' I/O Pin For LED

' Program Code
' ---------------------------------------------------------------------------

Start:
  TRIS_C        =           %10111111       ' Set I/O Pin Directions

Main:
  LED = PIR                                 ' Make LED Follow PIR
  GOTO Main
```