LIGA DE ENSINO DO RIO GRANDE DO NORTE
CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE
ESPECIALIZAÇÃO EM REDES DE COMPUTADORES

LAILTON MONTENEGRO DE VASCONCELOS

MONITORAMENTO DE TEMPERATURA E UMIDADE DE DATA CENTER UTILIZANDO O ARDUINO E O SISTEMA ZABBIX

NATAL/RN
2017

LAILTON MONTENEGRO DE VASCONCELOS

MONITORAMENTO DE TEMPERATURA E UMIDADE DE DATA CENTER UTILIZANDO O ARDUINO E O SISTEMA ZABBIX

Trabalho de Conclusão de Curso apresentado ao Centro Universitário do RN, como requisito final para obtenção do título de Especialista em Redes de Computadores.

Orientador: Prof. M.Sc. Aluizio Ferreira da Rocha Neto

NATAL/RN

2017

Catalogação na Publicação – Biblioteca da UNI-RN
Setor de Processos Técnicos

Vasconcelos, Lailton Montenegro de.
Monitoramento de temperatura e umidade de data center utilizando o Arduino e o Sistema Zabbix / Lailton Montenegro de Vasconcelos. – Natal, 2017.
36 f.

Orientador: Prof. M.Sc. Aluízio Ferreira da Rocha Neto.
Monografia (Especialização em Redes de Computadores) – Centro Universitário do Rio Grande do Norte.

1. Data Center – Monografia. 2. IoT – Monografia. 3. Arduino. Zabbix – Monografia. I. Rocha Neto, Aluízio Ferreira da. II. Título.

RN/UNI-RN/BC CDU 004.72

LAILTON MONTENEGRO DE VASCONCELOS

MONITORAMENTO DE TEMPERATURA E UMIDADE DE DATA CENTER UTILIZANDO O ARDUINO E O SISTEMA ZABBIX

Trabalho de Conclusão de Curso apresentado ao Centro Universitário do RN, como requisito final para obtenção do título de Especialista em Redes de Computadores.

Aprovado em:_____/_____/_____.

BANCA EXAMINADORA

_____

Prof. Aluizio Ferreira da Rocha Neto

Orientador

_____

Professor Convidado 1

Membro

_____

Professor Convidado 2

Membro

RESUMO

A Segurança da Informação e a alta disponibilidade tem sido um grande desafio para as empresas no mundo, seja ela pequena, média ou grande. Todos estão preocupados em manter seus dados protegidos e sempre disponíveis. Umas das áreas da Segurança da Informação que se preocupa com a alta disponibilidade, é a área da refrigeração ou climatização de Data Center. Não menos importante que outras áreas da segurança, ela deve se preocupar em manter a temperatura ideal para os servidores, sempre constante. Por isso, monitorar este ambiente é primordial para o bom funcionamento. Existem muitas ferramentas no mercado, porém, com um custo bastante elevado, o que dificulta a implementação em pequenas e médias empresas. Com o surgimento da tecnologia *Internet of Things* (IoT) percebe-se que é possível implementar uma gama de sensores em dispositivos ligados à Internet, sensores estes capazes de coletar informações precisas e enviar estas informações da maneira que for necessária. Como uma solução de baixo custo, este trabalho implementa um sensor baseado em Arduino capaz de monitorar a temperatura e a umidade de um Data Center e enviar estes dados ao sistema de monitoramento Zabbix, permitindo o monitoramento proativo de todo o ambiente do Data Center, inclusive a partir da tela de um celular.

Palavras-chave: Data Center. IoT. Arduino. Zabbix.

ABSTRACT

Information Security and high availability has been a major challenge for companies around the world, whether small, medium or large. Everyone is concerned about keeping their data safe and always available. One of the Information Security areas that is concerned with high availability is the data center cooling or air conditioning area. No less important than other security areas, it should bother to maintain the ideal temperature for servers always constant. Therefore, monitoring this environment is paramount for proper operation. There are many tools on the market, however, with a very high cost, which makes it difficult to implement in small and medium enterprises. With the emergence of Internet of Things (IoT) technology, it is realized that it is possible to implement a range of sensors on Internet-connected devices, sensors that can collect accurate information and, in a way that is necessary, send this information. As a low-cost solution, this work implements an Arduino-based sensor capable of monitoring the temperature and humidity of a Data Center and sending this data to the Zabbix monitoring system, enabling proactive monitoring of the entire Data Center environment, including from the screen of a cell phone.

Keywords: Data Center. IoT. Arduino. Zabbix.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Quando se fala em Segurança da Informação veem-se logo em mente vários fatores como: Proteção de dados por firewall, antivírus, políticas de segurança, Pentest, campanhas de segurança, engenharia social, etc.

O controle e manutenção da temperatura e umidade em um ambiente data center são tão importantes quanto a manutenção de um firewall, e estão diretamente ligados ao negócio da empresa, e na disponibilidade da informação.

Existem várias tecnologias no mercado, sistemas capazes de implementar um monitoramento robusto, porém, são sistemas geralmente com um custo elevado e as vezes, necessárias mudanças no layout para passagem de fios e de complexidade elevada para implementação.

Foi buscando soluções de baixo custo, flexíveis, de fácil desenvolvimento e implementação, que iniciou-se as pesquisas nas tecnologias de *Internet of Things* (IoT). Atualmente no mercado, uma dessas tecnologias de maior evidencia é o Arduino. Assim, foi escolhida esta plataforma para o desenvolvimento de um projeto de monitoramento de temperatura e umidade de Data Center.

Além do Arduino como coletor da informação, foi utilizado o Zabbix como ferramenta de monitoramento, que ajuda a organizar os dados coletados em gráficos e gerar gatilhos caso haja alguma alteração que possa prejudicar o bom funcionamento do Data Center.

Em geral, o relacionamento entre as duas ferramentas se dará através do protocolo SNMP, e fará com que se tenha um sistema capaz de monitorar a temperatura e umidade do Data Center, visualizar em tempo real os dados coletados através de um ambiente web e a implementação de gatilhos, para o caso de a variação da temperatura interna do data center alcance níveis não sadios para o funcionamento de servidores e dispositivos ali presentes.

1.1 OBJETIVOS

1.1.1 Geral

O objetivo deste projeto é implementar o protocolo SNMP no Arduino para que ele sirva como um repositório de informações de sensores. Em nosso caso,

usaremos sensores de temperatura e umidade para monitoramento de um ambiente de data center, e o sistema de monitoramento Zabbix, como coletor das informações contidas nestes sensores.

1.1.2 Específicos

Para atingir o objetivo geral, as seguintes etapas foram seguidas na confecção deste documento:

- Escolha do Arduino como tecnologia IoT;
- Implementação do protocolo SNMP na placa Arduino;
- Integração do Arduino com o sistema de monitoramento Zabbix;
- Implementação em uma empresa como estudo de caso;

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma. O capítulo 2 apresenta o referencial teórico, contendo os conceitos, técnicas e protocolos utilizados deste projeto. O capítulo 3 apresenta um estudo de caso, apresentando o cenário pós instalação do sistema de monitoramento de temperatura e umidade em um data center e também a forma que o zabbix mostra o resultado das coletas dos sensores, e o capitulo 4 as considerações finais.

2 REFERENCIAL TEÓRICO

2.1 DATACENTER

2.1.1 Conceitos básicos

O termo da língua inglesa Data Center, é designado ao Centro de Processamento de Dados (CPD), local este onde são armazenados todos os dados e informações de uma empresa. No mundo, as grandes empresas investem milhares de dólares em datacenters do tamanho de fazendas, que sejam capazes de armazenar milhares de terabytes por dia.

O Data Center é basicamente um local, ou sala, devidamente preparada para receber diversos servidores, com bancos de armazenamento de dados, os quais processam e mantém uma enorme quantidade de informação. A estrutura de um datacenter consiste em Racks, geralmente de 40U. O "U" significa unidade de altura, é a medida padrão utilizada em rack. A altura de 1U é aproximadamente de 4,5 cm, e a maioria dos equipamentos de redes estruturadas seguem estas medidas segundo Veras (2009). É de extrema importância que exista um piso elevado, com espaço para cabeamento abaixo e sala climatizada (com ar-condicionado em temperatura constante), para resfriar os computadores, que deve ser totalmente protegida e monitorada (POR DENTRO..., 2012).

2.1.2 Aplicação

Segundo Veras (2009), Datacenter é o elemento central de processamento e armazenamento dos dados e da informação. Processa e armazena os dados e informações que suportam os processos organizacionais. Analisando este trecho abordado pelo professor, podemos pensar que a o datacenter passa a ser a coluna vertebral de uma empresa cuja sua importância atinge diretamente os negócios. Vale destacar os milhões de investimentos que as grandes empresas como Google, Facebook, Microsoft entre outras, nos levando a acreditar no quão importante é para uma empresa este ambiente. Em outro trecho o professor fala que “O datacenter é, na verdade, um provedor de serviços para aplicações, consequentemente, para os

processos de negocio. Dele depende o nível de serviço que o usuário recebe na ponta do processo” (VERAS, 2009).

2.2 INTERNET DAS COISAS (*INTERNET OF THINGS* – IOT)

2.2.1 Conceitos básicos

Segundo a Associação Brasileira de Internet das coisas “A Internet das Coisas” (*Internet of Things*) é um termo genérico usado para descrever um próximo passo na evolução da Internet. Enquanto a primeira fase da rede pode ser interpretado como uma combinação de rede interconectada de documentos de hiper – textos e uma rede de aplicações (considerando estas como sendo blogs, webmails, sites de redes sociais, etc.), um dos próximos passos é um Internet mais potente de objetos "inteligentes" - ou "coisas", que será acessível aos seres humanos e entre si por meio de conexões de rede. Isto é, a chamada Internet das Coisas” (ABINC, 2017).

De acordo com o Dave Evans, em um artigo publicado pela Cisco® A IoT (Internet of Things, Internet das coisas), algumas vezes referida como a Internet dos objetos, mudará tudo, inclusive nós mesmos. Isso pode parecer uma declaração arrojada, mas considere o impacto que a Internet já teve na educação, na comunicação, nos negócios, na ciência, no governo e na humanidade. Claramente, a Internet é uma das criações mais importantes e poderosas de toda a história humana. Agora, considere que a IoT representa a próxima evolução da Internet, dando um grande salto na capacidade de coletar, analisar e distribuir dados que nós podemos transformar em informações, conhecimento e, por fim, sabedoria. Nesse contexto, a IoT se torna bem importante (EVANS, 2011).

Estamos entrando em uma evolução onde todas as “coisas” estão acessando de algum modo na internet, com a funcionalidade de gerenciamento, ajudando o ser humano a resolver situações pré-programadas, sem a interversão do mesmo. As maquinas se comunicarão entre si, e escolherão o que deve ser feito diante de determinada situação.

2.2.2 Aplicação

São infinitas as possibilidades de aplicação, porém, podemos citar alguns exemplos usados no conceito de casas inteligentes. O simples fato de integração das lâmpadas de casas com o celular é um conceito de IoT. Onde o usuário poderá ligar ou desligar a lâmpada como lhe convir. Os ar-condicionados, tvs e demais objetivos que tenham acesso através de infravermelho, bluetooth ou outras tecnologias de acesso remoto, também estão sendo bastante usados. Através de um portal online você pode ser capaz de controlar todos estes objetos.

Um exemplo de quando a máquina poderá tomar uma decisão, sem que haja intervenção do homem é quando aplicamos essas soluções para regar plantas. Podemos configurar os sensores para monitorar a umidade do ambiente ou do solo, e caso o sistema, que foi pré-configurado, perceba que a umidade esta baixa, ele automaticamente decide que é hora de regar as plantas.

2.2.3 História

De acordo com Adriano Lucas Balaguer a origem do nome Internet das Coisas é atribuída a Kevin Ashton. O termo foi o nome de uma apresentação feita por ele em 1999 na empresa Procter & Gamble (P&G). Dez anos depois, num artigo publicado através do RFID Journal, ele referência a apresentação e cita o que é tido por muitos como a definição de IoT (TRIBUNA..., 2014).

De acordo com o Cisco Internet Business Solutions Group (IBSG), a IoT é o momento exato em que foram conectados à Internet mais "coisas ou objetos" do que pessoas. Em 2003, havia aproximadamente 6,3 bilhões de pessoas vivendo no planeta e 500 milhões de dispositivos conectados à Internet.3 Ao dividir o número de dispositivos conectados pela população mundial, descobrimos que existia menos de um (0,08) dispositivo por pessoa. Com base na definição do Cisco IBSG, a IoT não existia em 2003, pois o número de itens conectados era relativamente pequeno considerando que dispositivos ubíquos, como smartphones, estavam sendo apresentados. Por exemplo, Steve Jobs, CEO da Apple, não revelou o iPhone até 9 de janeiro de 2007 na conferência Macworld (EVANS, 2011).

2.3 ARDUINO

2.3.1 Conceitos básicos

Segundo Simon Monk (2013) Arduino é uma pequena placa de microcontrolador, contendo diversos terminais que permitem a conexão com dispositivos externos, como motores, relés, sensores e outros. Como o projeto da placa é aberto, existem inúmeras placas compatíveis de baixo custo. Outras placas acessórias, denominadas Shields, são essenciais para expandir a conexão do Arduino. Por exemplo, a placa shield Ethernet, usado para conectar a placa Arduino com rede Ethernet.

Em outras palavras, o Arduino é um minicomputador, que com a utilização de componentes de entrada e saída, pode ser programado para executar inúmeras tarefas, de uma maneira simplificada.

De acordo com Michael McRoberts (2011) A maior vantagem do Arduino sobre outras plataformas de desenvolvimento de microcontroladores é a facilidade de sua utilização. Pessoas leigas, ou com pouco conhecimento podem elaborar um pequeno projeto em um curto espaço de tempo.

2.3.2 Aplicação

A principal finalidade do Arduino num sistema é facilitar a prototipagem, implementação ou emulação do controle de sistemas interativos, a nível doméstico, comercial ou móvel. Com ele é possível enviar ou receber informações de basicamente qualquer sistema eletrônico, como identificar a aproximação de uma pessoa e variar a intensidade da luz do ambiente conforme a sua chegada. Ou abrir as janelas de um escritório de acordo com a intensidade da luz do sol e temperatura ambiente.

2.3.3 Historia do Arduino

O projeto iniciou-se na cidade de Ivrea, Itália, em 2005, com o intuito de interagir em projetos escolares de forma a ter um orçamento menor que outros

sistemas de prototipagem disponíveis naquela época. O site oficial do Arduino é o www.arduino.cc. Hoje existem inúmeros modelos de placas Arduino, conforme a seguinte lista dos principais: (Fonte www.arduino.cc)

- ARDUINO UNO
- ARDUINO MEGA 2560
- ARDUINO LEONARDO
- ARDUINO DUE
- ARDUINO MEGA ADK
- ARDUINO NANO
- ARDUINO PRO MINI
- ARDUINO ESPLORA

2.4 ZABBIX

2.4.1 Conceitos básicos

O Zabbix é uma ferramenta *open-source* de implementação do protocolo *Simple Network Management Protocol* (SNMP) de monitoramento de redes, servidores e serviços, visando o monitoramento da disponibilidade e qualidade dos serviços prestados em redes. A arquitetura Zabbix e a flexbilidade dos módulos permitem que a mesma seja utilizada de inúmeras formas como exemplo: on/off, acompanhamento de desempenho de aplicações, Serviços e outros.

A ferramenta de monitoramento de redes Zabbix oferece uma interface 100% Web para administração e exibição de dados. Os alertas do sistema de monitoramento Zabbix podem ser configurados para utilizar vários métodos de comunicação, como SMS, e-mail e abertura de chamados em sistemas de helpdesk. O sistema permite ainda que ações automáticas como, por exemplo, restart de serviços sejam executados a partir de eventos.

Existem basicamente três componentes. São eles: Zabbix Server, Zabbix Proxy e Zabbix Agent.

Zabbix Server é o componente central da ferramenta, é nele que chegará as informações coletadas, onde serão tratadas, processadas e apresentadas em

relatórios customizados.

Zabbix Proxy Funciona como uma ponte entre o Agent e o Server. É uma parte opcional da implantação do zabbix, mas que auxilia para um controle centralizado de locais remotos, filiais, coletando os dados e enviando ao Zabbix Server;

Zabbix agente é instalado nos hosts e utilizando métricas comuns, como por exemplo, servidores Windows, ele enviará ao server informações da memória, processador, capacidade de discos e etc.

2.4.2 Aplicação

A ferramenta Zabbix pode ser usada para monitorar praticamente tudo que esteja na rede. Sua simplicidade, interface atraente e eficácia são adjetivos que justificam a utilização dessa poderosa ferramenta. Como o Zabbix pode monitorar praticamente tudo em uma rede, ele se torna um grande aliado dos administradores de rede, permitindo uma rápida reação para problemas em servidores ou ativos de redes. O monitoramento constante da Saúde da rede, servidores e Ativos de redes, acaba tornando os seus administradores cada vez mais proativos, identificando um problema em sua raiz, eliminando a possibilidade de um possível incidente.

2.4.3 História

De acordo com o seu fórum oficial, Zabbix.com, sua história teve início em 1998 por Alexei Vlasishev, no qual seria utilizado apenas em seu local de trabalho, logo após a primeira versão não estável lançada em 2001 (v.1.0aphal). Em 2004 foi lançado a versão instável (v.1.0), em 2005 foi fundado a Zabbix SIA Company que presta suporte técnico e comercial do produto, em 2012 foi criado o Zabbix Japão LLC, ao longo dos anos foram surgindo novas versões e aprimorando mais recursos. Há pouco tempo a Zabbix lançou a versão 2.2.4 que contém uma imensa lista de *features*.

Atualmente a Zabbix SIA tem 11 parceiros no Japão que promovem e fornecem produtos e serviços Zabbix para as empresas locais, incluindo Bancos, Companhias de Seguros, Telecomunicações, Manufatura, Transportes e Empresas de Logística, Educação e organizações de Assistência Médica de forma ativa. O

principal objetivo do Zabbix Japão LLC será envolver novos parceiros e apoiar o já existente, para ajudá-los a chegar a ainda mais empresas em diversos setores. A subsidiária também será a realização de atividades de marketing e publicidade, tornando a marca Zabbix ainda mais reconhecível no mundo.

3 IMPLEMENTAÇÃO DO PROJETO

Inicia-se o projeto, enxergando a necessidade de monitoramento em um data center. No Data Center usado para o projeto, existe um monitoramento de temperatura, fazendo uso do Zabbix para administrar estas informações, porém, usa-se a temperatura interna dos servidores, podendo levar a vários falso-positivo. Quando uma das máquinas requeriam mais do processador, ela atingia níveis de temperaturas muito alto, que ativavam as triggers do Zabbix, enviando alertas para os responsáveis. Diante do exposto, foi preciso elaborar uma solução rápida, de fácil implementação física e com uma ótima precisão, e que fosse possível integração com o Zabbix.

Foi feito uma pesquisa na linha das tecnologias IoT e a melhor opção encontrada, levando-se em conta o custo benefício, foi no Arduino UNO, visto na Figura 1.

Figura 1 – Placa Arduino Uno usada neste projeto

Fonte: https://cdn.sparkfun.com//assets/parts/6/3/4/3/11021-01c.jpg (2017)

Inicialmente o pensamento foi em usar tela de LCD, fazer um case bem moderno, usar vários sensores, colocar alguns led's para indicar determinadas situações, acesso web para mostrar os dados e ainda enviar as informações coletadas via SNMP para o zabbix.

Ao iniciar o desenvolvimento, percebeu-se que o projeto em si, estava perdendo a sua funcionalidade, já que o principal objetivo era coletar as informações de temperatura e umidade do data center. E a complexidade da implementação do protocolo SNMP no Arduino, estava maior do que foi esperado. Ao perceber o tempo perdido, com implementações com LCD, LED's, sistemas Web, decidiu-se abortar

esta parte e deixar o projeto o mais simples possível. Deixando-o barato e focando na problemática de implementação do protocolo SNMP.

Com a ideia de implementar o projeto em partes decidiu-se dividir o projeto da seguinte forma: Primeiro, focar no objetivo central do projeto “monitoramento de temperatura e umidade”, as implementações posteriores, ou upgrades, seriam implementando gradativamente. O tema desta monografia, se prende a esta primeira parte do projeto.

3.1 COMPONENTES

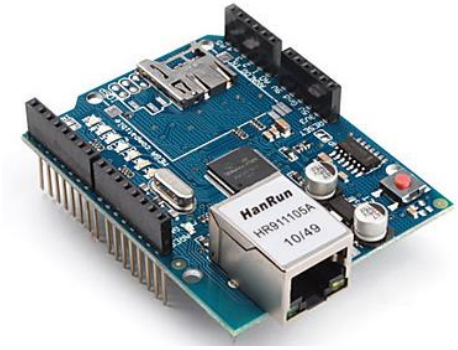
Nesta seção serão descritos os componentes utilizados para a construção do modelo de sistema.

3.1.1 Arduino Uno

Para este projeto alguns itens essenciais foram escolhidos. O primeiro deles, é o próprio Arduino. A opção do Arduino UNO (Figura 1), foi motivado, por que a quantidade de pinos analógicos seria suficiente, e no momento, para uma rápida implementação seria o mais viável, como também o baixo custo no mercado.

Após a escolha do Arduino, foi preciso decidir qual a melhor opção para conexão na rede LAN para coleta das informações. Apesar de várias opções de conexões, como wifi, Bluetooth, radio frequência, e etc, mas, novamente, focando na praticidade e simplicidade do projeto, foi escolhido o Ethernet Shield W5100, conforme visto na Figura 2.

Figura 2 – Placa Ethernet Shield W5100 usada neste projeto.



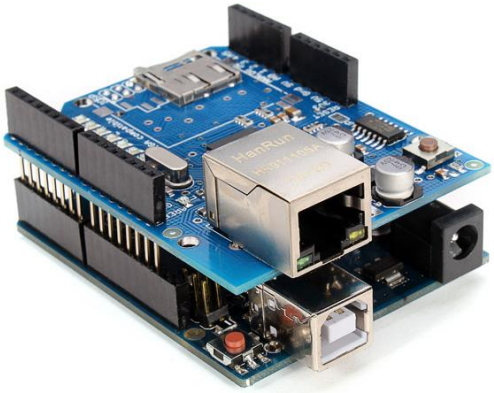
Fonte: http://luthortronics.com.br/wp-content/uploads/2015/10/qnfgjz1339666028684.jpg (2017)

3.1.2 Ethernet Shild W5100

O Ethernet Shield W5100 permite que a Arduino se conecte a rede LAN, através de uma porta RJ45 com velocidade de 10/100Mbps. Na verdade, é um componente que acoplado ao Arduino, conforme imagem 3, possibilita a conexão de rede. Pode ser usado como cliente ou servidor. Talvez fosse usado como servidor, se na programação do sistema, as coletas dos sensores fossem expostas em uma página HTML, ou seja, se fosse criado uma página web, contendo as informações das coletas dos sensores. Neste projeto, o desenvolvimento foi feito usando a placa como cliente, pois não haveria necessidade de uma página web, já que as informações serão mostradas diretamente no zabbix.

A Figura 3 demonstra como fica quando se acopla a Shild Ethernet no Arduino.

Figura 3 – Arduino com o Ethernet Shield W5100 acoplada.



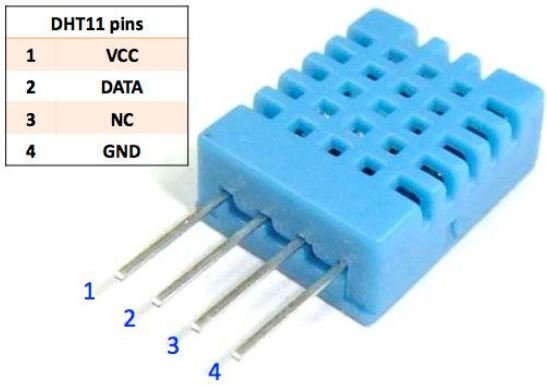
Fonte: http://img.banggood.com/images/2014/xiemeijuan/11/SKU099516SKU066313/SKU083549C.jpg (2017)

3.1.3 Sensor DHT11

Para coletar as informações, a opção mais viável, foi o sensor DHT11, pois em um só componente, ele nos fornece informações de temperatura e umidade.

Este sensor inclui um componente medidor de umidade e um componente NTC para temperatura, ambos conectados a um controlador de 8-bits. O interessante neste componente é o protocolo usado para transferir dados entre o MCDU e DHT11, pois as leituras do sensor são enviadas usando apenas um único fio de barramento (AOSONG, 2017).

Figura 4 – Sensor de umidade e temperatura DHT11.



Fonte: AOSONG (2017)

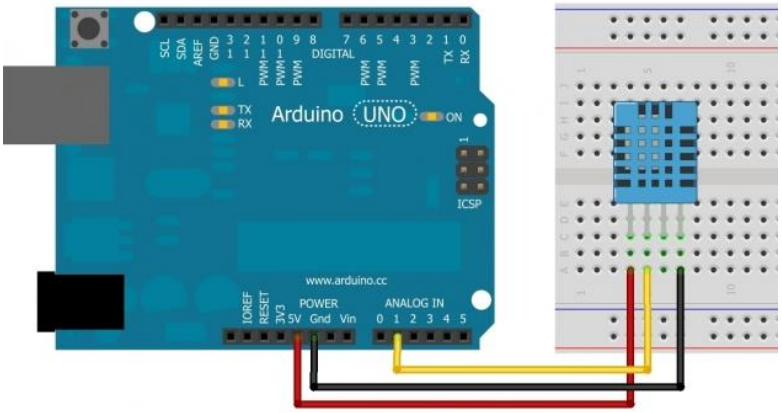
Algumas especificações importantes:

- Faixa de medição de umidade: 20 a 90% UR
- Faixa de medição de temperatura: 0º a 50ºC
- Precisão de umidade de medição: ± 5,0% UR
- Precisão de medição de temperatura: ± 2.0 ºC

3.2 MONTAGEM

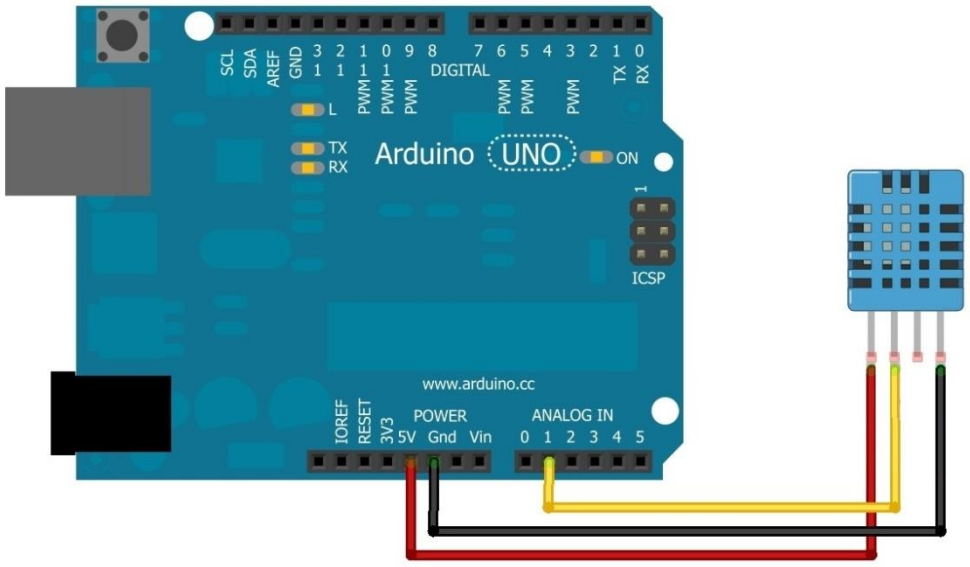
Após a escolha dos componentes, deu-se início a montagem. O circuito eletrônico ficou muito simples, sem a necessidade de uma placa para colocação do sensor DHT11. Primeiramente foi usado uma protoboard, que é uma placa de ensaio ou matriz de contato, com furos e conexões condutoras para montagem de circuitos elétricos experimentais, em seguida, excluiu-se a necessidade da protoboard e foi feito uma ligação direta do sensor DHT11 com os pinos digitais e eletrônicos, tornando a implementação ainda mais simples. Nas Figuras 5 e 6 os dois exemplos.

Figura 5 – Montagem do circuito em Protoboard



Fonte: Material desenvolvido pelo autor (2017).

Figura 6 – Circuito montado sem a necessidade da Protoboard



Fonte: Material desenvolvido pelo autor (2017).

3.3 PROGRAMAÇÃO DO ARDUINO

3.3.1 Bibliotecas

Uma biblioteca é um trecho de software que fornece funcionalidade específica a um programa, como por exemplo a capacidade de escrever em um display de LCD ou de controlar a posição de um servomotor. O uso de uma biblioteca simplifica o desenvolvimento de aplicações, pois o código da biblioteca já está pronto, e só

precisa ser incorporado ao programa em desenvolvimento para que suas funções possam ser acessadas e utilizadas pelo desenvolvedor. Assim, podemos estender o uso do Arduino incorporando bibliotecas específicas durante o desenvolvimento. Algumas das bibliotecas mais usuais foram usadas para programar o Arduino. Porém, uma delas é de suma importância para o funcionamento do sistema de monitoramento. Trata-se da biblioteca **Agentuino**, a qual implementa o protocolo SNMP, que é necessário para se comunicar com o Zabbix na transmissão dos dados coletados pelos sensores.

Para este projeto, foi preciso usar as seguintes bibliotecas de código:

- DHT.h - usada para leitura dos dados do sensor DHT11;
- Ethernet.h - usada na comunicação com a interface Ethernet do *shield*;
- SPI.h - usada para comunicação com o *shield* Ethernet; e
- Agentuino.h - usada para implementação do protocolo SNMP.

3.3.2 Implementação SNMP utilizando a biblioteca Agentuino

Uma das exigências deste projeto foi a implementação da coleta e disponibilidade dos dados via protocolo SNMP, visando a não dependência de ferramentas específicas. Criando uma MIB com as informações coletadas pelos sensores, seria possível integrar com qualquer ferramenta de monitoramento padrão do mercado compatível com SNMP, como é o caso do Zabbix, escolhido neste projeto.

A problemática do projeto, que demandou mais tempo, foi justamente a implementação deste protocolo, no caso, a biblioteca Agentuino que é uma biblioteca SNMP para a plataforma Arduino. O conceito desta biblioteca ainda é pouco discutido, tanto na web como entre desenvolvedores, por este motivo, foi difícil encontrar assuntos relacionados, que servisse de base para o desenvolvimento do projeto.

Foi preciso fazer muitas pesquisas, e debater o assunto com alguns desenvolvedores, que também não tinham muitas informações a respeito. O projeto chegou a parar algumas vezes esperando respostas de fóruns, e outros especialistas. Cogitou-se mudar a metodologia, aplicando uma *Application Program Interface* (API) do Zabbix para coletar os dados, porém, o projeto ficaria fortemente dependente do software do Zabbix.

A biblioteca Agentuino pode ser encontrada no seguinte endereço da Internet: <https://code.google.com/archive/p/agentuino/>. Trabalhando na base da tentativa e erro, foi preciso enxugar o código da biblioteca ao máximo, adequando as necessidades do projeto. Como pode ser visto no trecho de código a seguir, é necessária a criação da MIB SNMP para uso das funções da biblioteca Agentuino. Esta MIB está definida nas linhas 2 a 43, além dos objetos que representam as informações de temperatura (linha 29) e de umidade (linha 30). Na sequência, que vai da linha 31 a 37, as outras informações da MIB, como descrição, nome, contato e etc.. O trecho de código entre as linhas 44 e 71, foi todo preservado do código de exemplo de uso da biblioteca Agentuino, pois está diretamente ligado ao funcionamento da conexão e de seus componentes.

Trecho de código de uso da biblioteca Agentuino.

```
 1  # Agentuino
 2  # RFC1213-MIB OIDs
 3  # .iso (.1)
 4  # .iso.org (.1.3)
 5  # .iso.org.dod (.1.3.6)
 6  # .iso.org.dod.internet (.1.3.6.1)
 7  # .iso.org.dod.internet.mgmt (.1.3.6.1.2)
 8  # .iso.org.dod.internet.mgmt.mib-2 (.1.3.6.1.2.1)
 9  # .iso.org.dod.internet.mgmt.mib-2.system (.1.3.6.1.2.1.1)
10  #...system.sysDescr(.1.3.6.1.2.1.1.1) read-only RO - DisplayString
11  const static char sysDescr[20] PROGMEM = "1.3.6.1.2.1.1.1.0";

12  #...system.sysObjectID (.1.3.6.1.2.1.1.2) RO - ObjectIdentifier
13  const static char sysObjectID[20] PROGMEM = "1.3.6.1.2.1.1.2.0";

14  #...system.sysUpTime (.1.3.6.1.2.1.1.3) RO - TimeTicks
15  const static char sysUpTime[20] PROGMEM = "1.3.6.1.2.1.1.3.0";

16  #...system.sysContact (.1.3.6.1.2.1.1.4) RW - DisplayString
17  const static char sysContact[20] PROGMEM = "1.3.6.1.2.1.1.4.0";

18  #...system.sysName (.1.3.6.1.2.1.1.5) RW - DisplayString
19  const static char sysName[20] PROGMEM = "1.3.6.1.2.1.1.5.0";

20  #...system.sysLocation (.1.3.6.1.2.1.1.6) RW - DisplayString
21  const static char sysLocation[20] PROGMEM = "1.3.6.1.2.1.1.6.0";

22  #...system.sysServices (.1.3.6.1.2.1.1.7) RO - Integer
23  const static char sysServices[20] PROGMEM = "1.3.6.1.2.1.1.7.0";

24  # Arduino defined OIDs
25  # .iso.org.dod.internet.private (.1.3.6.1.4)
26  # .iso.org.dod.internet.private.enterprises (.1.3.6.1.4.1)
27  # ...enterprises.arduino (.1.3.6.1.4.1.36582)
28  # ...arduino.value.valA0-A5 (.1.3.6.1.4.1.36582.3.1-6) RO - Integer
29  const static char temp[24] PROGMEM = "1.3.6.1.4.1.36582.3.1.0";  # RO
30  const static char umidade[24] PROGMEM = "1.3.6.1.4.1.36582.3.2.0"; # RO
```

```
31  const static char locDescr[20] = "Agentuino";
32  const static char locObjectID[20] = "1.3.6.1.4.1.36582";
33  const static uint32_t locUpTime = 0;
34  const static char locContact[20] = "lailton.montenegro@gmail.com";
35  const static char locName[20] = "LAILTON MONTENEGRO - Agentuino";
36  const static char locLocation[20] = "Natal, RN";
37  const static int32_t locServices = 7;
38  int loctemp = 0;
39  int locumidade = 0;

40  uint32_t dispMillis = 0;

41  char oid[SNMP_MAX_OID_LEN];
42  SNMP_API_STAT_CODES api_status;
43  SNMP_ERR_CODES status;


44  void pduReceived()
45  {
46    SNMP_PDU pdu;
47    Serial.println("UDP Packet Received...");
48    api_status = Agentuino.requestPdu(&pdu);
49    if ((pdu.type == SNMP_PDU_GET || pdu.type == SNMP_PDU_GET_NEXT ||
50         pdu.type == SNMP_PDU_SET) && pdu.error == SNMP_ERR_NO_ERROR &&
          api_status == SNMP_API_STAT_SUCCESS ) {
51      pdu.OID.toString(oid);
52      Serial.print("OID = ");
53      Serial.println(oid);

54      if ( pdu.type == SNMP_PDU_SET ) {
55        status = SNMP_ERR_READ_ONLY;
56      } else if ( strcmp_P(oid, sysDescr ) == 0 ) {
57        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locDescr);
58      } else if ( strcmp_P(oid, sysUpTime ) == 0 ) {
59        status = pdu.VALUE.encode(SNMP_SYNTAX_TIME_TICKS, locUpTime);
60      } else if ( strcmp_P(oid, sysName ) == 0 ) {
61        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locName);
62      } else if ( strcmp_P(oid, sysContact ) == 0 ) {
63        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locContact);
64      } else if ( strcmp_P(oid, sysLocation ) == 0 ) {
65        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locLocation);
66      } else if ( strcmp_P(oid, sysServices) == 0 ) {
67        status = pdu.VALUE.encode(SNMP_SYNTAX_INT, locServices);
68      } else if ( strcmp_P(oid, sysObjectID) == 0 ) {
69        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locObjectID);
70      } else if ( strcmp_P(oid, temp) == 0 ) {
71        status = pdu.VALUE.encode(SNMP_SYNTAX_INT, loctemp);
72      } else if ( strcmp_P(oid, umidade) == 0 ) {
73        status = pdu.VALUE.encode(SNMP_SYNTAX_INT, locumidade);
74      } else {
75        status = SNMP_ERR_NO_SUCH_NAME;
76      }
77      pdu.type = SNMP_PDU_RESPONSE;
78      pdu.error = status;
79      Agentuino.responsePdu(&pdu);
80    }
81    Agentuino.freePdu(&pdu);

82  }
```

Trecho de código do void setup(). Com destaque na linha 6 e na condição if.

```
1 void setup()
2 {
3  Serial.begin(9600);
4  Ethernet.begin(mac, ip, subnet);
5  dht.begin();
6
7  api_status = Agentuino.begin(); //inicialização do pacote Agentuino
8
9  if ( api_status == SNMP_API_STAT_SUCCESS ) { //Condições para
inicialização do Pacote Agentuino
10    Agentuino.onPduReceive(pduReceived);
11    delay(10);
12    Serial.println("SNMP Agent Initiated...");
13    return;
14  } else
15  {
16  delay(10);
17  Serial.print("SNMP Agent failed!");
18  }
19
20 }
```

Trecho de código do void loop() com destaque a linha 7.

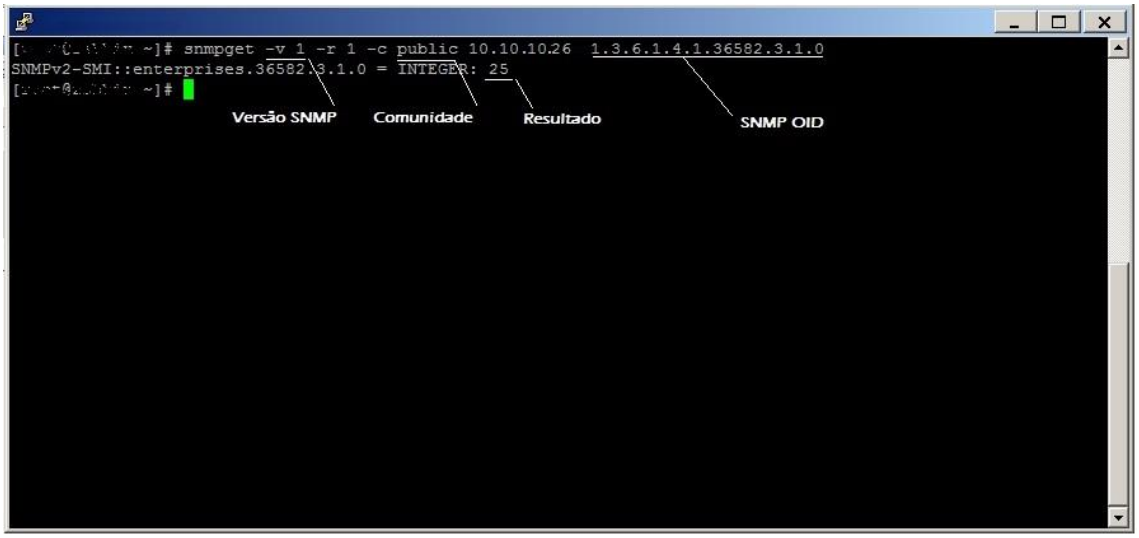
```
1 void loop()
2 {
3
4  int u = (dht.readHumidity()); // variavel para umidade
5  int t= (dht.readTemperature()); // variavel para temperatura

7  Agentuino.listen(); //identificador se existe solicitações SNMP
8
9    loctemp = t;
10   locumidade = u;

}
```

As informações de versão SNMP e da comunidade, estão configuradas no arquivo base de importação da biblioteca Agentuino e por padrão a versão SNMP é a 1 e a comunidade é *public*. Com estas informações, mais as informações das chaves OID, que estão nas linhas 29 e 30 conforme código acima, foi feito a primeira consulta, utilizando o SNMPGet do Linux, que trouxe os resultados das coletas dos sensores, conforme a Figura 7, que mostra o resulta da coleta de temperatura.

Figura 7 – Consulta da temperatura usando o sistema SnmpGet no Linux.



Fonte: Material desenvolvido pelo autor (2017).

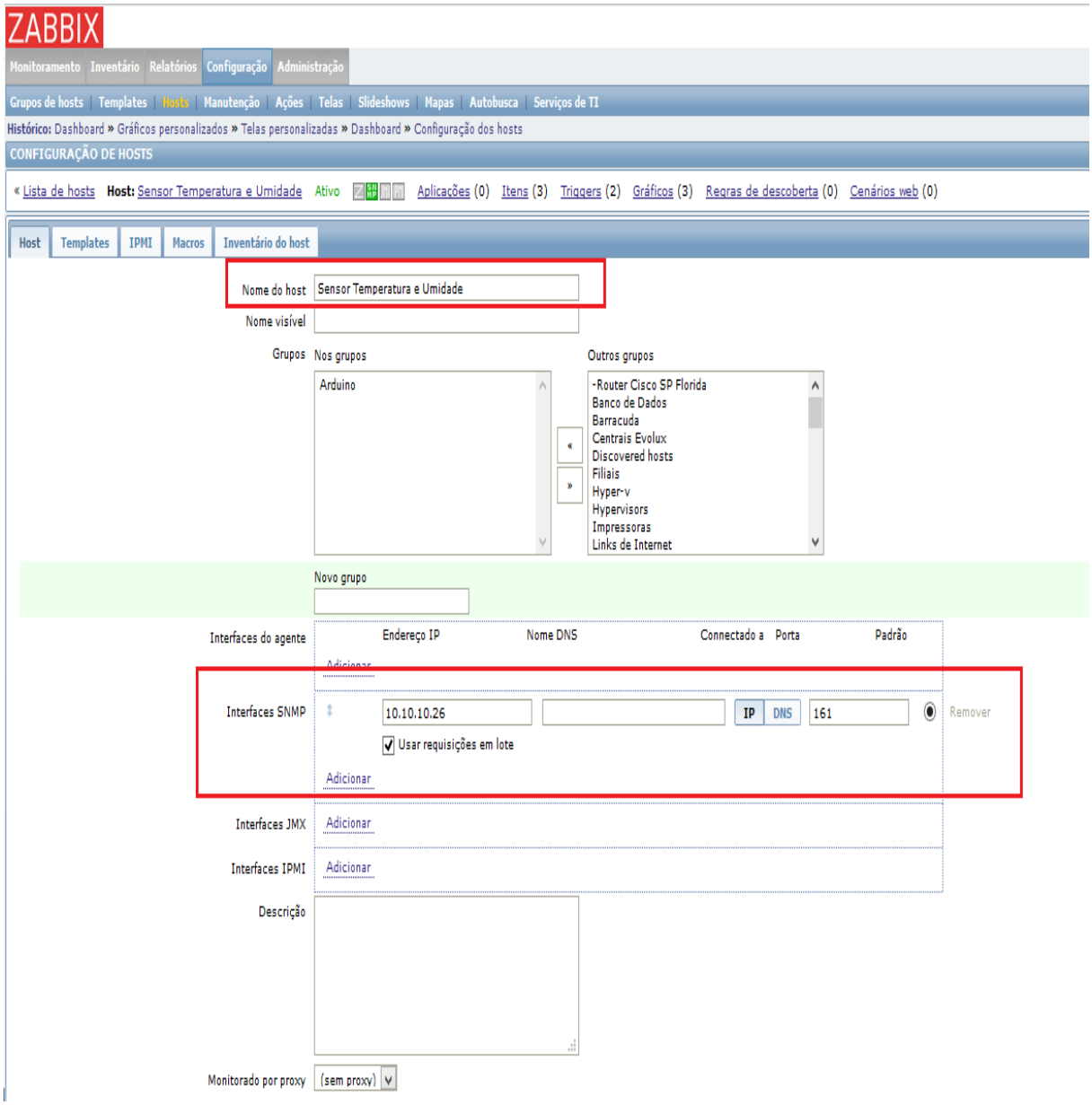
3.4 INTEGRAÇÃO COM O ZABBIX

Após conclusão da etapa de configuração do pacote Agentuino e construção da MIB, e consequentemente, coletar as informações de temperatura e umidade, a integração com o Zabbix foi a etapa final do projeto. Conforme dito anteriormente, o objetivo da criação de uma MIB no Arduino, foi justamente para dar a possibilidade de integração com qualquer ferramenta com o propósito de monitoramento. O Zabbix esta foi a ferramenta escolhida para implementação do projeto.

Logo abaixo tem a descrição dos processos para configuração do Zabbix. A configuração é padrão para esta versão do Zabbix usado neste projeto, ou seja, não será usado este item como um passo a passo, mas sim, como demonstração da configuração para coleta das informações da MIB em questão.

Primeiro passo foi a criação do host, informando um nome para o host, o tipo de interface, em nosso caso *interface SNMP*, o IP e a porta de comunicação SNMP, conforme Figura 8.

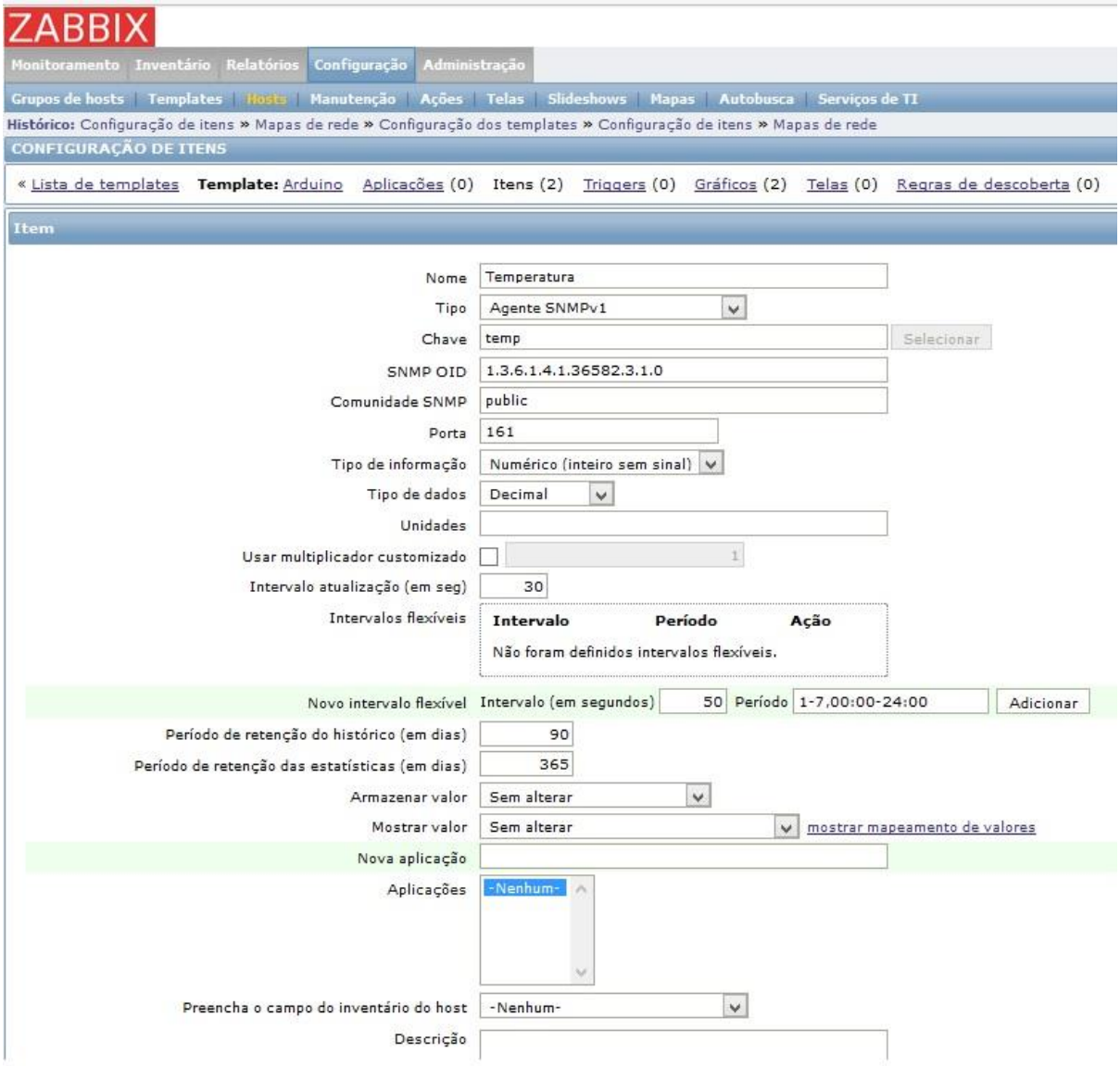
Figura 8 – Criação do Host no Zabbix.



Fonte: Material desenvolvido pelo autor (2017).

O 2º passo foi criar a *template* que chamamos de *Arduino*, Conforme a Figura 9, foram criados dois itens, um para cada sensor, Temperatura e Umidade.

Figura 9 – Itens temperatura e umidade no Zabbix.

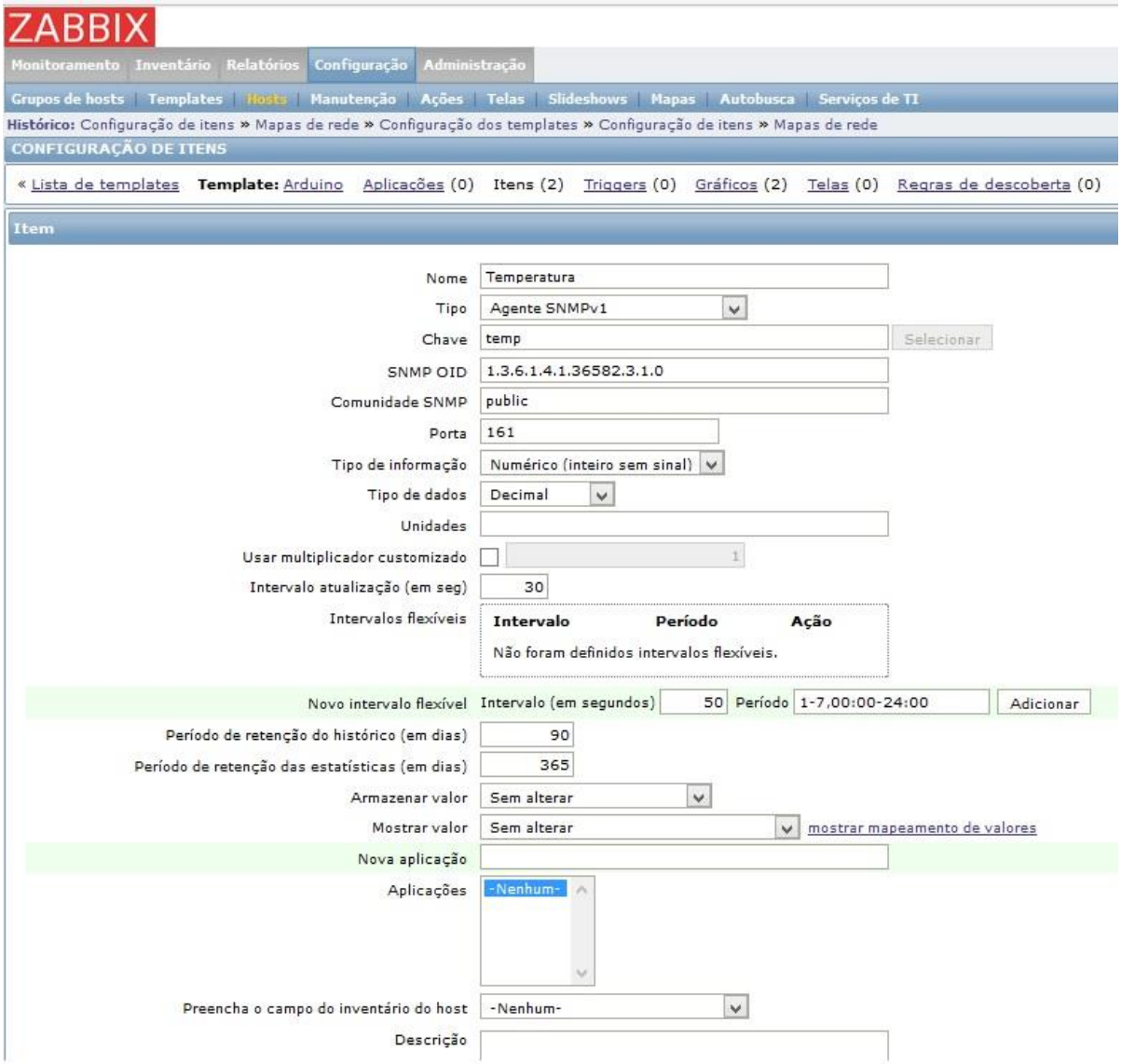


Fonte: Material desenvolvido pelo autor (2017).

As informações são as mesmas utilizadas para coletar os dados com o snmpget no Linux, com exceção da chave, que no zabbix foi preciso preencher. Estas chaves foram configuradas no código agentuino implementado no projeto. No **Trecho de Código da Biblioteca Agentuino** acima**,** pode ser visto nas linhas 29 e 30.

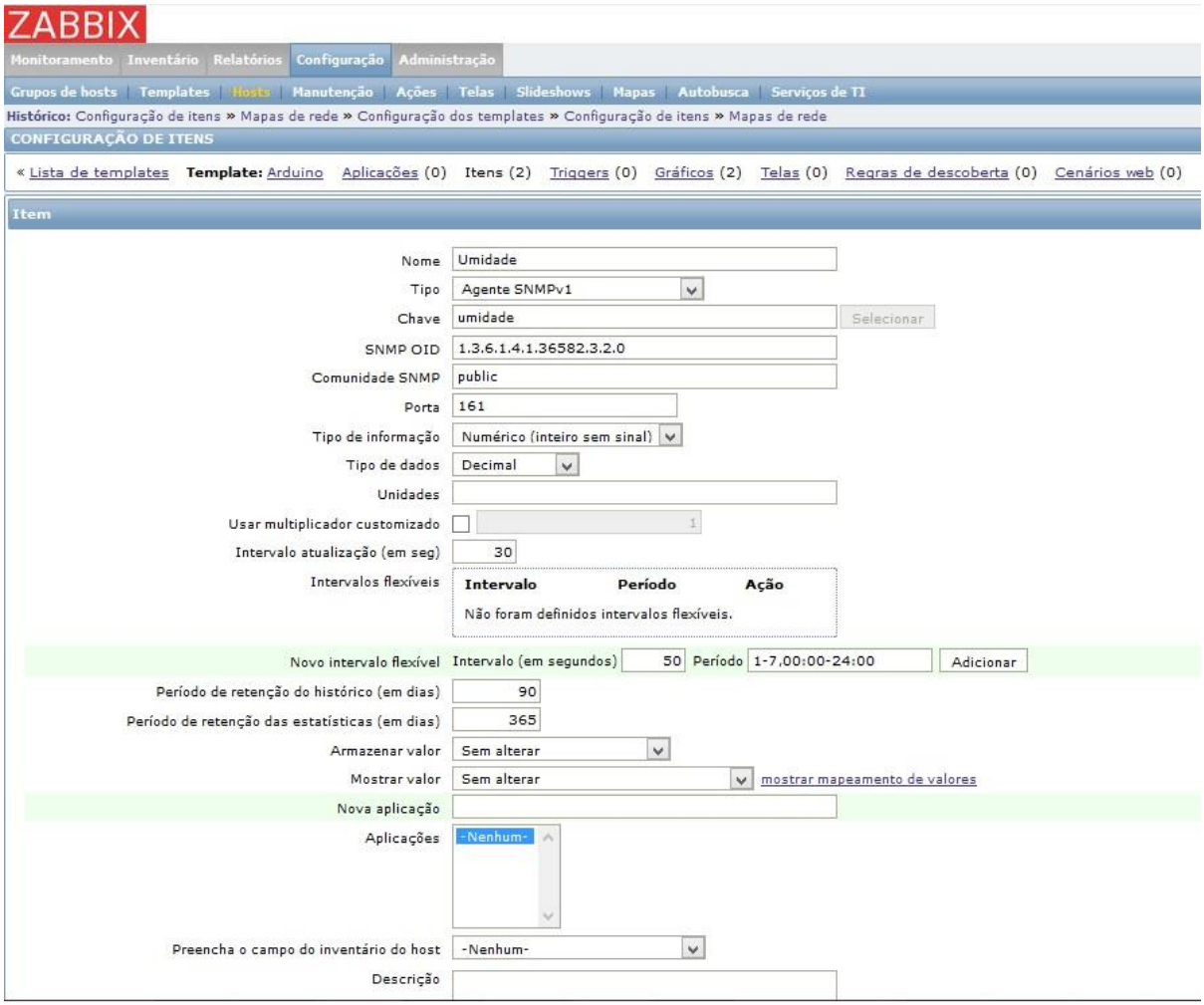
Nas Figuras 10 e 11 duas telas com as configurações dos itens temperatura e umidade respectivamente.

Figura 10 – Configuração do Item temperatura no Zabbix.



Fonte: Material desenvolvido pelo autor (2017).

Figura 11 – Configuração do Item umidade no Zabbix.

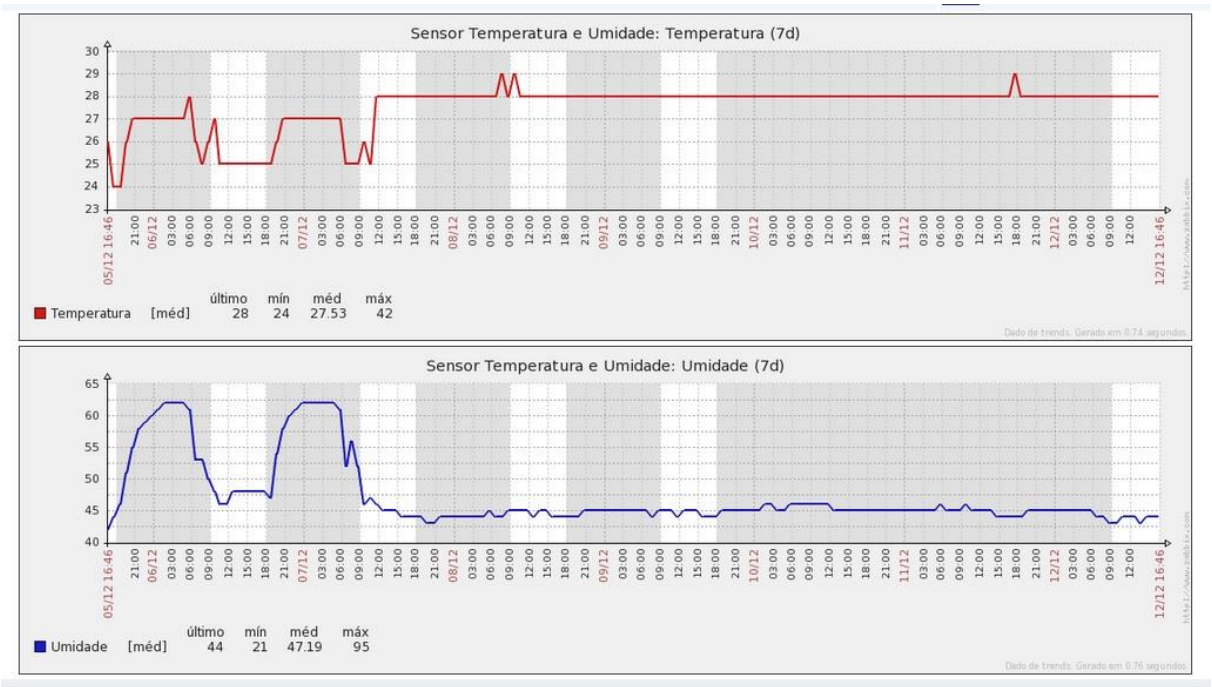


Fonte: Material desenvolvido pelo autor (2017).

3.4.1 Resultado do Monitoramento Zabbix

Após o fim das configurações, os resultados são expostos em números, por isso, para um acompanhamento mais usual, utilizamos a opção de gráficos e apresentamos em um monitor para acompanhamento da equipe responsável, conforme Figura 12.

Figura 12 – Resultado da coleta de temperatura e umidade exposto em gráficos no Zabbix.



Fonte: Material desenvolvido pelo autor (2017).

3.5 UTILIZAÇÃO PRÁTICA DA FERRAMENTA APÓS IMPLEMENTAÇÃO

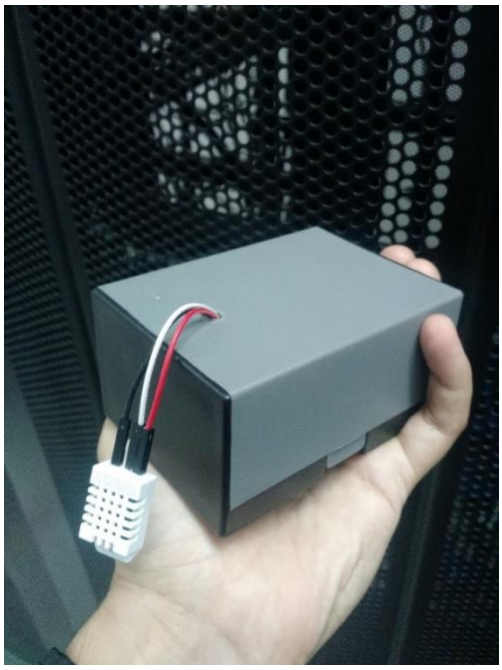
Com o término das configurações, foi instalado o equipamento em um data center, e disponibilizado o monitoramento para que a equipe de redes tenha acesso e possa visualizar os dados coletados.

Implementou-se também, alguns alertas caso a temperatura atinja determinado valor. Neste caso, colocamos um alerta para caso a temperatura fique acima de 35 Cº por 1 minuto, e a umidade se mantenha acima de 60% por mais de 5 minutos ou abaixo de 30% pelo mesmo tempo.

Estes alertas são considerados como críticos, por isso, são enviadas mensagens via SMS, e e-mails, configurados no próprio Zabbix.

Conforme visto na Figura 13, optou-se em colocá-lo em um case de plástico, para deixa-lo com uma aparência mais profissional, como também, evitar o contato da placa direto com o ambiente. Já nas Figuras 14 e 15 podemos conferir o local que escolhemos para o monitoramento do Data Center. Este está localizado exatamente no meio do rack dos servidores.

Figura 13 – Projeto finalizado em case plástico com o sensor do lado externo da caixa

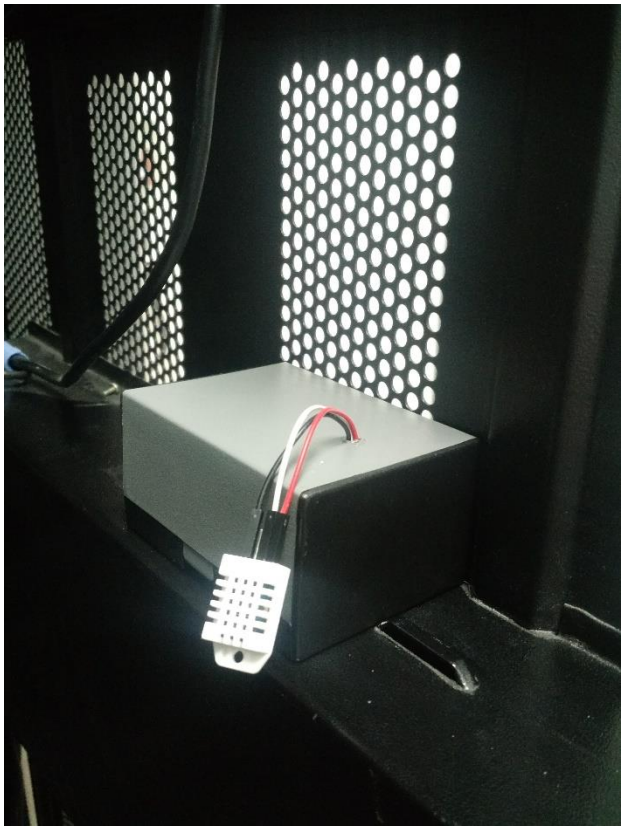


Fonte: Material fotografado pelo autor (2017).

Figura 14 – Local de instalação do sistema de monitoramento

Fonte: Material fotografado pelo autor (2017).

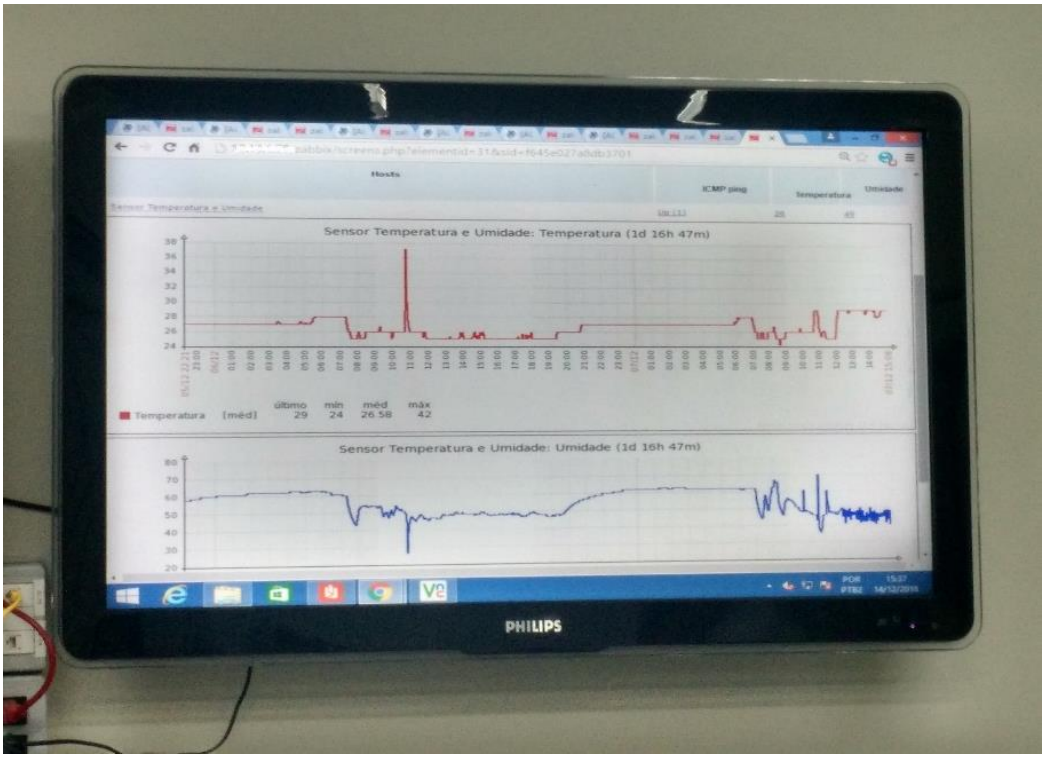
Figura 15 – Outra foto do local de instalação do sistema de monitoramento



Fonte: Material fotografado pelo autor (2017).

Para acompanhamento dos dados, as informações coletadas pelo Zabbix, são disponibilizados em uma TV, que fica disponível na sala do NOC (*Network Operation Center*) para a equipe de redes e infraestrutura, conforme Figura 16. Caso ocorra algum alerta, o Zabbix irá alterar a cor padrão da janela de monitoramento de temperatura e a equipe que acompanha esses dados diariamente, poderá tomar uma ação com maior rapidez.

Figura 16 – Gráficos coletados expostos em TV na sala do NOC



Fonte: Material fotografado pelo autor (2017).

4 CONSIDERAÇÕES FINAIS

O propósito principal deste trabalho foi de apresentar um sistema de baixo custo, confiável e de integração com qualquer outro sistema de monitoramento que trabalhe com o protocolo SNMP.

Um ponto de relevância para este projeto foi com relação ao custo benefício. Implementou-se um projeto de baixo custo, e de grande confiabilidade. Para se ter uma ideia, neste projeto o valor gasto foi de aproximadamente R$150,00. Comparando com sistemas mais robustos de grandes empresas nacionais, o projeto teve uma economia de mais de 98%. Outro ponto a se destacar no projeto foi a implementação do pacote agentuino, uma ideia ainda pouco discutido no mundo Arduino. O desenvolvimento deste projeto, a simplicidade do código e do circuito foi o maior objetivo. Espera-se que este projeto possa servir de base para outros ainda maiores.

Com relação ao efeito positivo na implementação, aplicou-se a uma placa, um sistema de monitoramento de temperatura e umidade, que, aliado ao Zabbix, e suas triggers, os administradores de redes e infra, tem a tranquilidade que serão avisados caso algo de errado ocorra com a refrigeração do datacenter, dando-lhes a possibilidade de um trabalho mais seguro e proativo.

Após o termino e implementação do projeto, observou-se que seria possível aumentar o nível e a qualidade do monitoramento, possibilitando a implementação de outros sensores, com o objetivo de trazer informações importantes. Alguns exemplos de sensores possíveis de implementação, como upgrade do projeto inicial seriam: sensores de fumaça, sensores de tensão, sensores de presença, de ruído e inúmeros outros, ou seja, a partir deste projeto inicial existem várias possibilidades de conexão com o Arduino, aumentando a complexidade do projeto, e tornando-o ainda mais completo.

Por fim, pode-se considerar que o projeto, em seu objetivo inicial, de criação de mib para o Arduino e integração com Zabbix foi atingido.

REFERÊNCIAS

AOSONG. **Temperature and humidity module:** DHT11 Product Manual. Disponível em: <http://img.filipeflop.com/files/download/Datasheet_DHT11.pdf>. Acesso em: 23 mar. 2017.

ARDUINO. Disponível em: <www.arduino.cc>. Acesso em: 23 mar. 2017.

ASSOCIAÇÃO BRASILEIRA DE INTERNET DAS COISAS (ABINC). **Plano nacional de IOT.** Disponível em: <http://www.abinc.org.br/>. Acesso em: 23 mar. 2017.

EVANS, Dave. A Internet das coisas: como a próxima evolução da Internet está mudando tudo. **Cisco Internet Business Solutions Group (IBSG),** abr. 2011. Disponível em: <http://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iot_ibsg_0411final.pdf>. Acesso em: 23 mar. 2017.

MCROBERTS, Michael. **Arduino básico.** São Paulo: Novatec, 2011.

MONK, Simon. **Programação com arduino:** começando com sketches. Porto Alegre: Bookman, 2013.

POR DENTRO DOS PRINCIPAIS DATACENTERS DA TERRA. 2012. Disponível em: <http://www.tecmundo.com.br/servidor/23963-por-dentro-dos-principais-datacenters-da-terra.htm>. Acesso em: 23 mar. 2017.

TRIBUNA DA BAHIA. **A internet das coisas:** das origens ao futuro. 2014. Disponível em: <http://www.tribunadabahia.com.br/2014/10/27/internet-das-coisas-das-origens-ao-futuro>. Acesso em: 23 mar. 2017.

VERAS, Manoel. **Data center:** componente central da infraestrutura de TI. Rio de Janeiro: Brasport, 2009.

ZABBIX: The Enterprise-class Monitoring Solution for Everyone. Disponível em: <Zabbix.com>. Acesso em: 23 mar. 2017.