UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

WASHINGTON JUNIOR FERREIRA

Monitoramento de Temperatura e Umidade em Racks Simples de Telecomunicação com Plataforma Arduino

Maceió - AL

2021

WASHINGTON JUNIOR FERREIRA

MONITORAMENTO DE TEMPERATURA E UMIDADE EM RACKS SIMPLES DE TELECOMUNICAÇÃO COM PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso submetido ao Curso de Sistemas de Informação do Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do Grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Lucas Benevides Viana de Amorim

Maceió - AL
2021

WASHINGTON JUNIOR FERREIRA

MONITORAMENTO DE TEMPERATURA E UMIDADE EM RACKS SIMPLES DE TELECOMUNICAÇÃO COM PLATAFORMA ARDUINO

Este Trabalho de Conclusão de Curso (TCC) foi julgado adequado para obtenção do Título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo Instituto de Computação da Universidade Federal de Alagoas.

Maceió, 01 de fevereiro de 2021.

_____

Prof. PETRÚCIO ANTÔNIO MEDEIROS BARROS, Msc.
Coordenador do Curso de Sistemas de Informação

Banca Examinadora:

_____

Prof. LUCAS BENEVIDES VIANA DE AMORIM
Orientador

_____

Prof. PETRÚCIO ANTÔNIO MEDEIROS BARROS, Msc.
IC/UFAL

_____

Prof. ALMIR PEREIRA GUIMARÃES, Dr.
IC/UFAL

_____

EVERTON CLEITON DE OLIVEIRA, Bsc.
NTI/UFAL

AGRADECIMENTOS

Em primeiro lugar a Deus, pois sem Ele nada seria possível. Que fez com que meus objetivos fossem alcançados durante o curso.

Aos meus pais, Jerôncio e Solange, por nunca terem medido esforços para me proporcionar um ensino de qualidade durante todo o meu período escolar.

A minha esposa e filho, Mara e Washington Filho, por estarem sempre ao meu lado, incentivando e dando apoio incondicional.

Aos meus irmãos, Willangy e Alex, e todos familiares, pelo companheirismo, pela cumplicidade e pelo apoio em todos os momentos delicados da minha vida.

Aos professores, pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

Ao meu orientador, Prof. Lucas Benevides, que conduziu o trabalho com paciência e dedicação, sempre disponível a compartilhar todo o seu vasto conhecimento.

Aos meus colegas de curso, com quem convivi durante os últimos anos, pelo companheirismo e pela troca de experiências que me permitiram crescer não só como pessoa, mas também como formando.

À instituição de ensino UFAL, essencial no meu processo de formação profissional, pela dedicação, e por tudo o que aprendi ao longo dos anos do curso.

Aos SGPTI do HUPAA/UFAL, local onde trabalho, e me autorizaram a fazer os testes em um ambiente de produção, que foram de grande utilidade para a elaboração deste trabalho.

Aos Analistas de Processos e Telecom, Antônio Fernando (HU-UFS) e Mário Cézar (MCO-UFBA) respectivamente, que me deram uma enorme força durante o curso, ajudando nas tarefas das disciplinas Algoritmo I e II.

Aos Analistas de Rede, Rodrigo Lotierzo (HU-UFS) e Caio Aragão (HUPAA-UFAL), o primeiro que me incentivou a realizar esse projeto, e o outro me ajudou na reta final do trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

AC	Corrente alternada (Alternating Current)
ASHRAE	American Society of Heating, Refrigerating and Air Conditioning Engineers (Sociedade Americana de Engenheiros de Aquecimento, Refrigeração e Ar Condicionado
CAD	Computer Aided Design (Desenho Assistido por Computador)
CDC	Container Data Center
CI	Circuito Integrado
DC	Corrente Contínua (Direct Current)
DIO	Distribuidor Interno Óptico
HD	Hard Disk (Disco Rígido)
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
HU	Hospital Universitário
HUPAA/UFAL	Hospital Universitário Prof. Alberto Antunes da Universidade Federal de Alagoas
HU-UFS	Hospital Universitário da Universidade Federal de Sergipe
ICSP	In Circuit Serial Programming (programação serial em circuito)
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
IP	Internet Protocol (Protocolo de Internet)
JSON	JavaScript Object Notation
LED	Light Emitting Diode (Diodo Emissor de Luz)
mA	Miliamperes
MAC	Media Access Control (Controle de Acesso de Mídia)
MHz	Milhão de Hertz
MIB	Management Information Base (base de informações de gerenciamento)
SNMP	Simples Network Protocol Manegement (Protocolo Simples de Gerenciamento de Redes)
SPI	Serial Peripheral Interface (interface periférica serial)
Telecom.	Telecomunicações
USB	Universal Serial Bus (Porta Serial Universal)
V.	Volts
XML	eXtensible Markup Language

RESUMO

Os racks de telecomunicações, onde são instalados os equipamentos de rede, na maioria das vezes ficam isolados em salas sem circulação de pessoas, e estão sujeitos a sofrer as mais diversas variações de temperatura e humidade, fazendo com que, de acordo com os manuais dos fabricantes, a vida útil desses equipamentos possivelmente seja reduzida, e consequentemente, dependendo do grau dessa variação, pode prejudicar os serviços de rede (internet e intranet) que dependem desses aparelhos. Foi vivenciando uma dessas variações de umidade, que danificou um switch, parando alguns serviços *web* de um hospital, a partir daí surgiu a ideia para resolução desse problema. O trabalho apresentado descreveu os passos executados para a construção de uma solução de baixo custo para o monitoramento de temperatura e umidade, direcionada para as pequenas empresas ou instituições que não dispõem de um capital financeiro mais elevado, para compra de uma solução mais robusta. Utilizou-se como base o Arduino, que é uma plataforma open-source de prototipagem eletrônica com *hardware* e *software* flexíveis e fáceis de usar, juntamente com um sensor e um *shield ethernet*. A solução foi testada em um ambiente de produção, instalada num rack simples de telecomunicação. O monitoramento foi feito através do software Zabbix, que é uma ferramenta de monitoramento distribuído e *Open Source*, que recebeu e armazenou as informações, gerando alertas aos usuários quando uma destas grandezas ultrapassou os limites estabelecidos, desta forma, assistindo o ambiente monitorado e observando se as condições do ambiente estão adequadas para o bom funcionamento dos equipamentos instalados no rack. Ao final do teste no ambiente de produção, a solução mostrou-se eficiente, cumprindo o objetivo do projeto, que seria uma solução barata e eficaz, desenvolvida valendo-se dos conceitos e tecnologias da plataforma Arduino.

Palavras-chave: Arduino. Zabbix. Temperatura e Umidade. Rack

ABSTRACT

Telecommunications racks, where network equipment is installed, are most often isolated in rooms with no circulation of people, and are subject to the most diverse variations in temperature and humidity, making it, according to the user manuals. manufacturers, the useful life of this equipment is likely to be reduced, and consequently, depending on the degree of this variation, it may harm the network services (internet and intranet) that depend on these devices. It was experiencing one of these humidity variations, which damaged a switch, stopping some hospital web services, that's where the idea for solving this problem came from. The work presented described the steps taken to build a low-cost solution for monitoring temperature and humidity, aimed at small companies or institutions that do not have a higher financial capital, to purchase a more robust solution. Arduino was used as a base, which is an open-source electronic prototyping platform with flexible and easy to use hardware and software, together with a sensor and an ethernet shield. The solution was tested in a production environment, installed in a simple telecommunication rack. The monitoring was done through the Zabbix software, which is a distributed and Open Source monitoring tool, which received and stored the information, generating alerts to users when one of these quantities exceeded the established limits, thus, watching the monitored environment and observing if the environmental conditions are adequate for the proper functioning of the equipment installed in the rack. At the end of the test in the production environment, the solution proved to be efficient, fulfilling the objective of the project, which would be a cheap and effective solution, developed using the concepts and technologies of the Arduino platform.

Keywords: Arduino. Zabbix. Temperature and Humidity. Rack

CAPÍTULO 1 - INTRODUÇÃO

Sabemos que equipamentos eletrônicos de uma forma geral, geram grandes quantidades de calor, em especial, aqui vamos enfatizar equipamentos de redes, pois estes trabalham em tempo integral e não podem ser desligados, isso faz com que aqueçam mais que equipamentos que passam períodos desligados. Dentre esses, vamos destacar os switches, que são responsáveis por interligar computadores de uma determinada empresa ou instituição, seja ela de grande ou pequeno porte, na rede mundial de computadores. Se eles sobreaquecerem podem chegar ao ponto de danificarem, causando prejuízos, da mesma forma que se a umidade relativa do ar for muito alta seu funcionamento pode ser prejudicado.

Problemas relacionados à alta umidade em equipamentos já foram presenciados no HU-UFS (Hospital Universitário da Universidade Federal de Sergipe), por exemplo, em que em uma determinada situação de chuvas intensas, um setor teve seu rack de equipamentos alagados, causando a queima dos equipamentos e a interrupção da conectividade e, portanto, de todos os serviços de TI dependentes da rede naquele setor. (LOTIERZO, 2016)

A ideia de fazer um sistema de baixo custo de monitoramento de temperatura e umidade com o Arduino, surgiu a partir daí, pois ao fazer uma pesquisa no mercado não encontramos uma solução simples, barata e de curto prazo para esse problema. Então, se tivéssemos um sistema em que fosse possível nos alertar sobre a umidade que estava subindo no local do rack de telecom., poderíamos assim ter evitado a danificação do equipamento, e desta forma não prejudicaria os serviços dos sistemas web no Hospital.

Visando o problema de monitorar esses dois dados essenciais para o bom funcionamento dos switches, que são a temperatura e a umidade, de racks simples em locais diversos e fora dos padrões recomendados, como é na maioria das pequenas e médias empresas e instituições, começamos procurar soluções focando no problema em si, para evitar que outros problemas do mesmo tipo ocorressem novamente. Hoje em dia ainda temos muitos problemas devido ao não monitoramento desses dados, assim fazendo com que atrapalhem ou mesmo interrompam os serviços de rede em funcionamento nesses locais.

Para o bom funcionamento dos equipamentos de rede, como switches, é necessário que eles trabalhem em temperatura e umidade recomendadas pelos fabricantes, de outra forma prejudicaria o trabalho das pequenas empresas e instituições que dependem desses equipamentos, para que seus serviços oferecidos sejam executados da melhor forma possível, ou seja, sem interrupções ou defeitos de rede. Partindo do pressuposto que, nem toda empresa

ou instituição dispõe de recursos financeiros suficientes para instalação de um equipamento de monitoramento profissional, que faça a análise desses dados e gerem alertas para que o problema seja evitado antecipadamente, sendo assim, pretende-se alcançar uma solução barata e eficaz, e com um sistema de monitoramento efetivo e de qualidade, provendo acesso em tempo real aos dados e com alertas que poderão evitar mau funcionamento desses equipamentos. Por fim, pretendemos responder com este trabalho a seguinte pergunta de pesquisa: É possível construir uma solução de baixo custo que seja eficaz para monitorar racks de telecomunicações em empresas e instituições de pequeno porte?

1.1 Objetivos

Esse trabalho visa demonstrar, que é possível construir uma solução de baixo custo, para monitorar racks de telecomunicações em empresas e instituições, e uma dessas soluções é com uma placa Arduino, um controlador *ethernet* e alguns sensores, juntamente com um software de monitoramento gratuito e de código aberto, que nesse caso usaremos o Zabbix, que é um software que monitora vários parâmetros da rede, dos servidores e da disponibilidade dos serviços. Utiliza-se de um mecanismo flexível de notificação que permite configurar alertas por e-mail para praticamente qualquer evento. (ZABBIX, 2020)

1.1.1 Geral

- Propor uma solução para monitoramento de temperatura e umidade em racks de telecom. utilizando a plataforma Arduino.

1.1.2 Específicos

- Implementar na plataforma Arduino, o sensor DHT11 que verifica as condições do ambiente, como temperatura e umidade;
- Integrar o Zabbix, que será a ferramentas de monitoramento.
- Monitorar o ambiente em tempo real, onde o rack de telecom. está instalado e confirmar com dados os resultados obtidos;

1.2 Soluções Existentes

No mercado existem várias soluções para monitorar temperatura e umidade, porém para grandes servidores em CDC ou Salas Cofres, mas com o custo muito elevado, tendo as seguintes características: Software completo de monitoramento e sensores de umidade e temperatura; monitor com conexão ethernet; e Display digital que possibilite a visualização das medições.

Um equipamento desse porte pode variar de valor, numa média entre dois a três mil reais, como pode ser visto em alguns pregões eletrônicos do governo, a exemplo do Pregão n.º 08/2018 do Ministério da Defesa - Comando da Aeronáutica. (COMPRASNET, 2018)

Outro dispositivo embarcado, encontrado no mercado, com a finalidade de monitorar ambientes, é o Poseidon2 3266, que suporta até 8 sensores, 4 detectores conectados a entradas digitais, podendo assim detectar fumaça, água, gás, vibrações, além de também monitorar temperatura e umidade. O dispositivo em questão pode ser configurado para enviar e-mail aos usuários registrados, bem como ser monitorado via SNMP (Simple Network Management Protocol) por alguma aplicação específica como o Zabbix ou Cacti. (HW GROUP, 2021)

CAPÍTULO 2 – FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será realizada a abordagem sobre a fundamentação teórica das tecnologias, equipamentos utilizados e suas condições climáticas ideais.

2.1 Equipamentos de telecomunicações

2.1.1 - *Switch*

O Switch é um importante equipamento, que possibilita a conexão na rede de computadores, ou seja, o switch realiza a conexão entre várias máquinas. É um dispositivo de interconexão, do tipo concentrador de rede, porém, ele divide a rede em domínios de colisão independentes, isso significa que os dados saem do dispositivo de origem e são encaminhados pelo switch apenas para o dispositivo de destino, sem que essas informações tenham que ser retransmitidas para todos os nós da rede. O *switch* está presente em todas as redes existentes atualmente. (ALECRIM, 2019)

A indústria trabalha basicamente com dois tipos de switches: ***switch* gerenciável** e ***switch* não gerenciável**, e podemos encontrar vários modelos com 8, 16, 24, 48 ou 96 portas, por exemplo. Os modelos não gerenciáveis são mais baratos e simples, e são usados em pequenas redes (como a de um pequeno escritório), que não exige muito tráfego de dados de diferentes tipos. Isso porque os switches não gerenciáveis são do tipo *plug and play* (plugue e use), podemos assim dizer: tudo que precisa fazer, é conectar a ele todos os computadores que fazem parte da rede, e com uma simples configuração nesses computadores, já começam a se "falarem". Não é possível fazer nenhuma configuração específica neles, a não ser ajustar um ou outro parâmetro ligado ao funcionamento da própria rede. Na Figura 1, temos a imagem de um *switch* não gerenciável. (ALECRIM, 2019)

Figura 1 - *Switch* não gerenciável - TP-Link TL-SG1016D 16 portas. Fonte: Modificado TP-Link (2020, p. 1)

Já em um switch gerenciável é diferente, Figura 2, pode se configurar vários parâmetros, para aumentar a segurança da rede, melhorar o fluxo de dados, priorizar tipos de tráfegos diferentes, entre outras configurações. Com ele pode-se ainda monitorar a rede, inclusive remotamente, através do protocolo SNMP (CASE, FEDOR, *et al.*, 1990), que

traduzindo a sigla para o português significa: protocolo simples de gerenciamento de redes, que é usado para transportar informações entre entidades gerenciadoras e agentes. (KUROSE e ROSS, 2010)

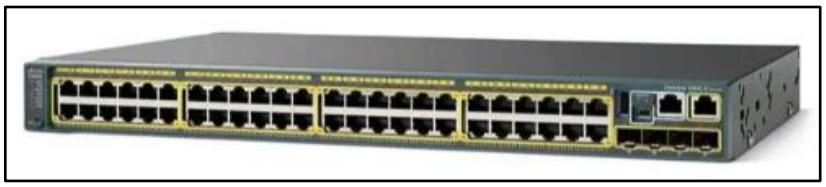

Figura 2 - Switch gerenciável - Cisco Catalyst 2960s 48 Portas. Fonte: Modificado de Cisco (2020, p. 1)

Os equipamentos de TI, como switches e servidores, independente do fabricante possui uma temperatura ideal, seja para maior eficiência, desempenho e aproveitamento da energia (COSENTINO, 2015). O modelo apresentado na Figura 2, de acordo com as especificações de instalação no manual do fabricante, orienta que a temperatura nas proximidades da unidade não pode ser superior a 45ºC. Se o switch for instalado em uma montagem fechada ou em vários racks, a temperatura nas proximidades da unidade pode ser superior à temperatura normal do ambiente da sala, e a umidade relativa do ar nas proximidades do *switch* não pode ser superior a 85%. (CISCO, 2012)

De acordo com a indústria, a temperatura ideal para uma sala de servidores onde estão concentrados todos os equipamentos TI, deve ser de 21ºC, com variação de 4ºC, isso porque vários equipamentos juntos concentram mais calor (COSENTINO, 2015). Mas para instalação de switches em racks simples de telecom. em locais variados, devemos seguir a orientação do fabricante, e qualquer medição acima dos dados informados, será necessário mais atenção e acompanhamento, pois a temperatura e umidade do local onde estão instalados os equipamentos estarão bem maior que a permitida, desta forma podendo danificá-los, causando a interrupção da conectividade de rede. Este acompanhamento é uma das tarefas de monitoramento apresentadas neste trabalho.

2.1.2 - Rack

Os racks são estruturas utilizadas para fixar e organizar switches, DIO’s, *hubs*, roteadores, *patch panels*, acessórios etc. A configuração física de um rack facilita a ventilação, organização e manutenção de cabos e demais acessórios, facilitando seu acesso. (TM TELECOM, 2020). São nesses racks que será instalado o monitoramento de temperatura e umidade. No mercado existem basicamente dois tipos de racks de vários tamanhos, os de parede e os de piso.

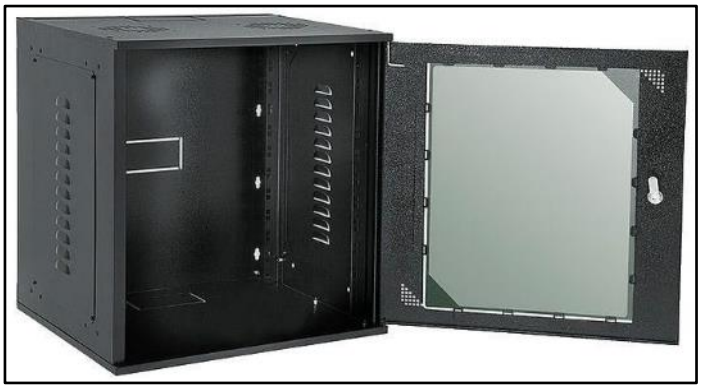

Figura 3 - Rack 19" Para Parede. Fonte: Modificado MKX e-commerce (2020, p. 1)

Figura 4 - Rack 19" Piso 20U. Fonte: Modificado de Layers Commerce (2020, p. 1)

“Rack Unit” ou "U" significa unidade de altura, é a medida padrão utilizada em racks de 19" (dezenove polegadas), seja ele de piso ou parede. A altura de 1U é aproximadamente de 4,5cm, e a maioria dos equipamentos de redes estruturadas seguem estas medidas, como exemplo: *patch panel*, switch, bandeja fixa, bandeja deslizante etc. (WBXRACKS, 2018)

2.2 - Condições climáticas ideais para os equipamentos instalados em racks

Sabemos que se a temperatura do ar ficar acima ou abaixo da especificação recomendada pelo fabricante do equipamento, pode resultar em falha e a redução da sua vida útil do mesmo. Igualmente a umidade, nesse caso, se há formação de umidade alta ou acumulação de eletricidade estática em pontos de umidade baixa podem ocasionar falhas de equipamento. (MARIN, 2011)

Apesar de haver grande discussão acerca de qual a temperatura ideal para operação de um data center, a recomendação da ASHRAE, é que a temperatura ideal na entrada de ar dos

equipamentos críticos de TI esteja entre 18°C e 27°C com umidade relativa do ar entre 40 e 55%. (FIGUEIREDO, 2017)

Como aqui não vamos tratar de data center, e sim de racks de telecom. instalados nos mais diversos ambientes, que não tem condições climáticas ideais, e às vezes não contam nem com um ar-condicionado de pequeno porte, então vamos usar como exemplo as especificações do fabricante do equipamento. Usaremos um Switch Cisco Catalyst 2960s (Figura 2), que nas suas especificações de ambiente operacional, a temperatura pode variar de 0ºC a 45ºC, e a umidade relativa do ar entre 5% e 85%. (CISCO, 2020)

2.3 - Plataforma Arduino

O Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software fáceis de usar. O Arduino pode sentir o estado do ambiente que o cerca por meio da recepção de sinais de sensores e pode interagir com os seus arredores, controlando luzes, motores e outros atuadores. As placas Arduino são capazes de ler entradas - luz em um sensor, um dedo em um botão ou uma mensagem no Twitter - e transformá-lo em uma saída - ativando um motor, ligando um LED ou publicando algo online. Com poucas linhas de comando, envia-se um conjunto de instruções ao microcontrolador na placa. Para fazer isso, usa-se a linguagem de programação Arduino, que foi baseada na linguagem *Wiring*, que é uma plataforma de prototipagem eletrônica de hardware livre, composta por uma linguagem de programação, um ambiente de desenvolvimento integrado (IDE) e um microcontrolador de placa única. (ARDUINO, 2018)

O Arduino IDE, é o software de código aberto que facilita a criação de códigos para carregá-los no microcontrolador. Por meio da programação em seu Ambiente de Programação Integrado (IDE - Integrated Development Environment), um software livre do próprio Arduino, que é utilizado para criação de códigos baseados na linguagem C, é possível designar programas de computadores a serem instalados em sua memória flash, fazendo com que o Arduino realize o conjunto de instruções que foram delineadas no software. (MCROBERTS, 2011)

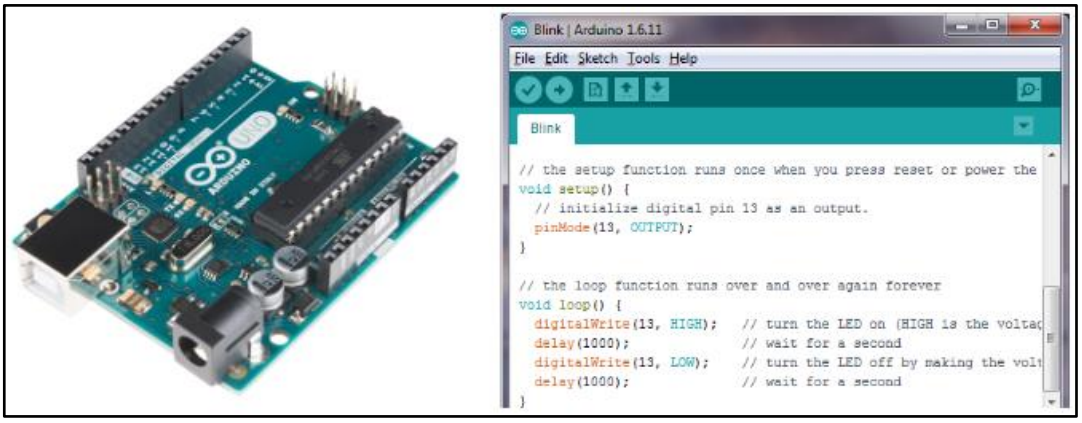

Figura 5 - Placa Arduino UNO Original da Itália à esquerda e Arduino IDE à direita. Fonte: Modificado de Robocore.net (2020, p. 2)

Os projetos desenvolvidos com o Arduino podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico. (ARDUINO, 2018)

Funcionando como um computador em miniatura, a plataforma Arduino é capaz de processar informações entre o próprio dispositivo e os componentes externos a ele conectados.

Na Figura 6 segue um exemplo de um projeto realizado na prática, em aula básica de Arduino, que foi desenvolvido na plataforma *Tinkercad*, que é uma ferramenta online de design de modelos 3D em CAD e também de simulação de circuitos elétricos analógicos e digitais, desenvolvida pela Autodesk (TINKERCAD, 2020). O pequeno projeto consiste basicamente em acender os *LEDs* de forma contínua das extremidades até o centro (*led* azul), e depois ir apagando no sentido contrário, do centro até as extremidades. Componentes usados no projeto: 01 Arduino UNO, 01 Protoboard (placa de prototipagem), 01 Bateria 9V, 09 *LEDs*, *09 Resistores 100 ohms* e 10 *Jumpers* (fios de conexão).

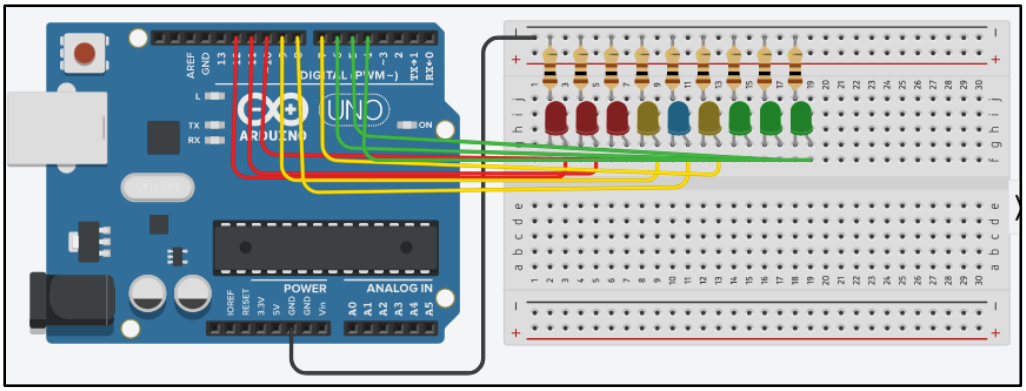

Figura 6 - Diagrama do projeto para acendimento contínuo de LEDs. Fonte: O Autor (TINKERCAD, 2020)

Na Figura 7, mostra-se de uma forma simples e com poucas linhas de códigos, que é fácil manipular as portas de entrada e saída do Arduino.

```
int pinLed[9] = {12,11,10,9,8,7,6,5,4};
int numLed;

void setup()
{
  int x;
  for (x = 0; x <= 8; x = x +1){
    pinMode(pinLed[x], OUTPUT);
  }
}

void loop()
{
  int aux = 8;
  int aux1 = 5;
  for (numLed = 0; numLed <=8; numLed = numLed + 1){
    digitalWrite(pinLed[numLed], HIGH);
    delay(100);
    digitalWrite(pinLed[aux], HIGH);
    delay(100);
    aux = aux -1;
  }

  for (numLed = 4; numLed >=0; numLed = numLed - 1){
    digitalWrite(pinLed[numLed], LOW);
    delay(100);
    digitalWrite(pinLed[aux1], LOW);
    delay(100);
    aux1 = aux1 +1;}

}
```

Figura 7 - Código fonte implementado no projeto. Fonte: O Autor

Nas Figuras 8 e 9, como já explicado anteriormente, mostra a sequência do funcionamento do projeto em fases, onde na 1ª fase todos os 09 Leds estão apagados, e dá início a execução do código, acendendo na 2ª fase os dois primeiros *leds* das extremidades, e os outros vão acendendo nas fases seguintes de forma contínua, e com um *deley* de 100 milissegundos (tempo de acendimento de um *led* para outro), até o *led* central(azul). Em seguida, a execução é feita de forma contrária, apagando primeiro o led central, seguindo até às extremidades com o mesmo tempo. Essa execução ficará em *loop* até que seja desligado o Arduino.

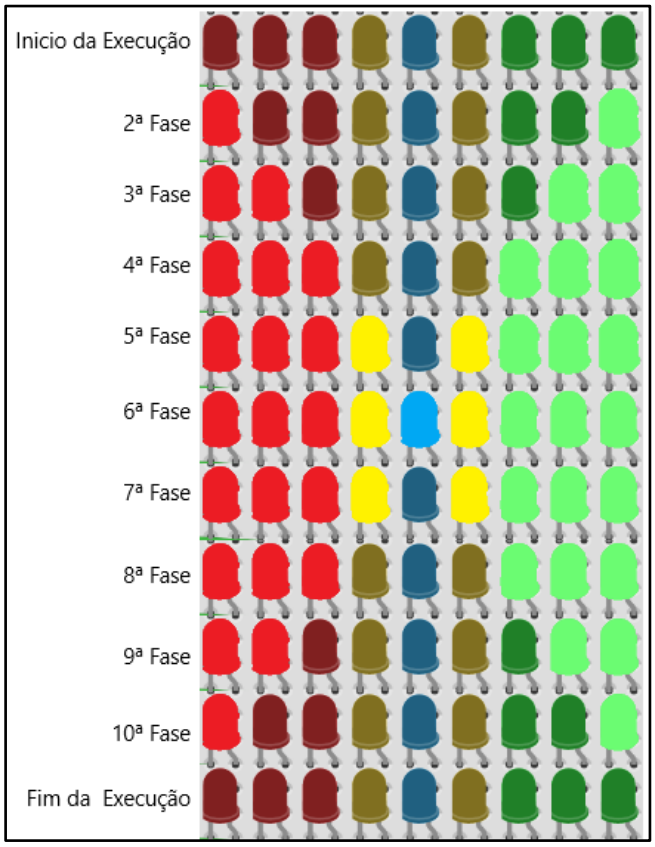

Figura 8 - Representa o resultado do projeto em funcionamento. Fonte: O Autor

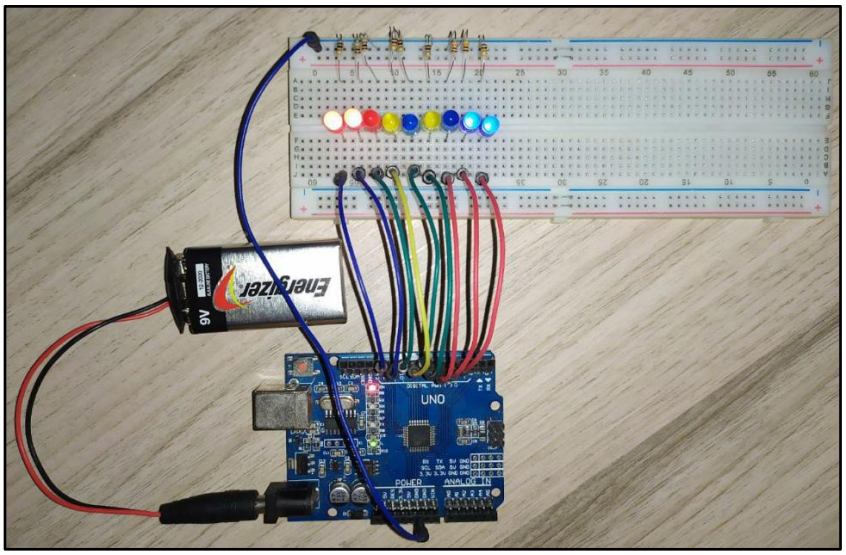

Figura 9 - Fotografia do projeto em funcionamento. Fonte: O Autor

Segundo McRoberts, uma das maiores vantagens da plataforma Arduino é sua arquitetura de hardware e software abertas, possibilitando a utilização livre por qualquer pessoa ou propósito, de seus códigos, esquemas ou projetos. (MCROBERTS, 2011)

2.4 - Arduino UNO

Existem diversos tipos de Arduino na atualidade, Arduino UNO, UNO R3, Mega, Leonardo, Pro Mini e outros. Para o nosso projeto usaremos o Arduino UNO, pelo custo-benefício e a sua facilidade de manipulação.

É uma placa baseada no microcontrolador ATmega328P (MICROCHIP, 2021), ele possui 14 pinos digitais de entrada e saída, 6 entradas analógicas, um cristal de quartzo de 16 MHz, conexão USB, um conector de energia, um botão ICSP e um botão de reset. Ele contém tudo para suportar o microcontrolador; simplesmente conecta-se a um computador por um cabo USB ou liga-se pelo seu conversor de corrente alternada para corrente contínua (*AC/DC - Alternating Current to Direct Current*) de 9v. (NATALMAKERS, 2015)

Na Figura 10, veremos os principais componentes do Arduino UNO.

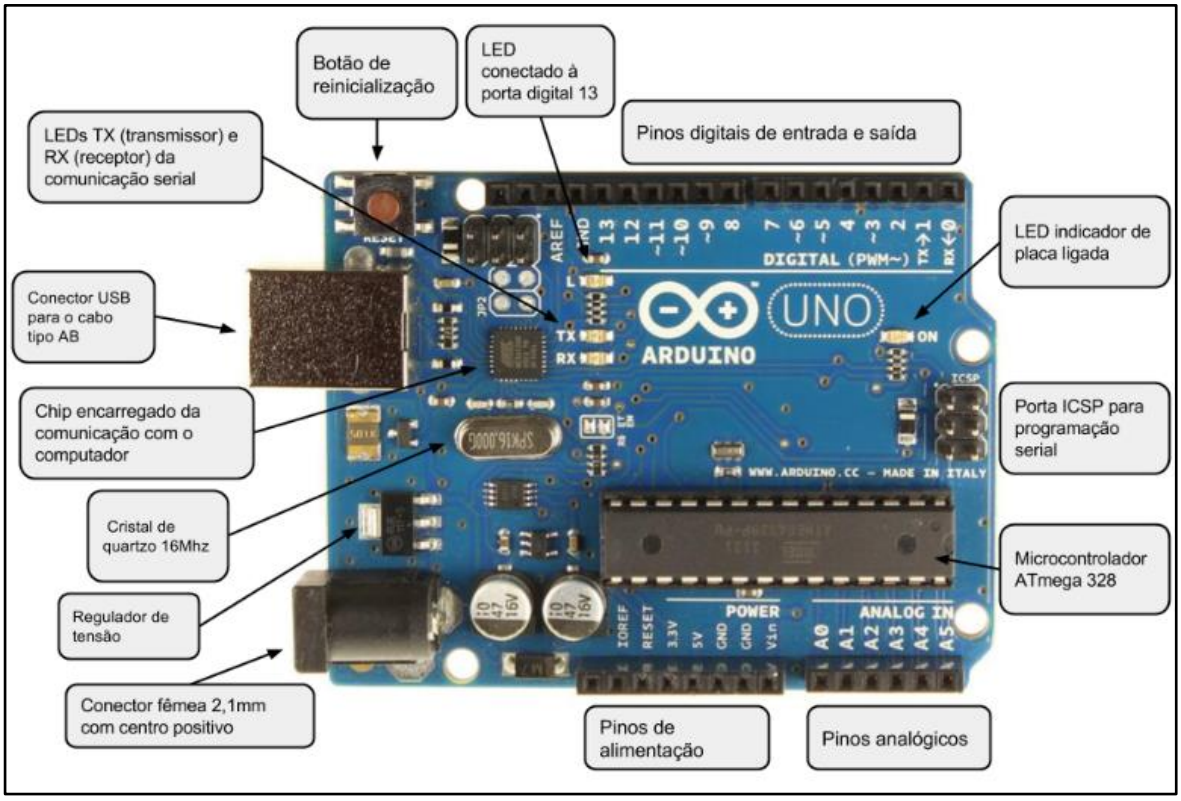

Figura 10 - Resumo dos componentes Arduino UNO. Fonte: Modificado de Natalmakers (2015, p. 1)

2.4.1 - Alimentação

O Arduino UNO pode ser alimentado pela conexão USB ou por uma fonte de alimentação externa. Na Figura 11, observa-se o circuito de alimentação.

Figura 11 - Alimentação do Arduino. Fonte: Modificado de Souza (2013, p. 2)

A alimentação externa é feita através do conector Jack com positivo no centro, onde o valor de tensão da fonte externa deve estar entre os limites de 6V a 20V.

A placa Arduino tem um CI *NCP1117*, que é o responsável pelo circuito regulador de tensão, que tem como objetivo regular a tensão de fonte externa de entrada, que é recomendado entre os valores de 7V a 12V (SOUZA, 2013). Tensões abaixo de 7V, fará com que o Arduino não funcione de forma satisfatória, acima de 12V o regulador de tensão poderá superaquecer e danificar a placa. Quando o cabo USB é ligado a um PC, por exemplo, a tensão não precisa ser estabilizada pelo regulador de tensão. Dessa forma a placa é alimentada diretamente pela USB. (SOUZA, 2013)

No circuito de alimentação, tem um botão *Reset* que serve para reiniciar o Arduino de forma física caso necessite, ou se a aplicação travar, ou terminar, basta você apertá-lo para que o programa reinicie.

Na figura 12, são exibidos os pinos de alimentação para conexão de *Shields* (os *shields* são placas que se encaixam ao Arduino para acrescentar funcionalidades de uma forma simples e confiável (QUADROS, 2021)), e módulos na placa Arduino UNO.

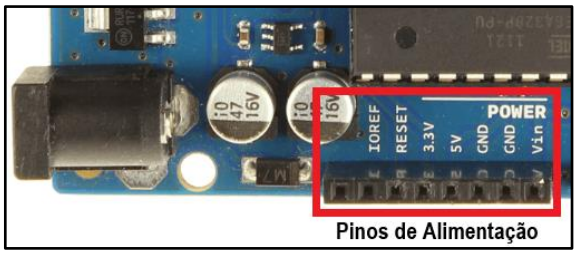

Figura 12 - Conectores de alimentação do Arduino. Fonte: O Autor

Segue abaixo a função de cada pino que são utilizados no Arduino UNO: (SOUZA, 2013)

- IOREF - Fornece uma tensão de referência para que *Shields* possam selecionar o tipo de interface apropriada, dessa forma *Shields* que funcionam com a placas Arduino que são alimentadas com 3,3V. podem se adaptar para ser utilizados em 5V e vice-versa.

- ➢ RESET - Pino de Reset do microcontrolador, tem a mesma função do botão Reset, só que é utilizado como uma função lógica, para um reset externo da placa Arduino.
- ➢ 3,3 V. - Fornece tensão de 3,3V para alimentação de *Shields* e módulos externos. Corrente máxima de 50 mA.
- ➢ 5V - Fornece tensão de 5V para alimentação de *Shields* e circuitos externos.
- ➢ GND - Pinos de referência, terra.
- ➢ VIN - Pino para alimentar a placa através de *Shields* ou bateria externa. Quando a placa é alimentada através do conector Jack, a tensão da fonte estará nesse pino.

2.4.2 - Comunicação

A interface de comunicação USB com o computador, é feita através de um microcontrolador ATMEGA16U2, como mostrado na Figura 13. Este microcontrolador é o responsável pela forma transparente como funciona a placa Arduino UNO, possibilitando o upload do código binário gerado após a compilação do programa feito pelo usuário. (SOUZA, 2013)

Figura 13 - Circuito de comunicação do Arduino. Fonte: O Autor

Nesse microcontrolador também estão conectados dois *LEDs* (TX, RX), controlados pelo software do microcontrolador, que indicam o envio e recepção de dados da placa para o computador, e um *LED* que indica que a porta digital está sendo usada. Esse microcontrolador possui um cristal de quartzo externo de 16 MHz, ele é uma estrutura em que os átomos se dispõem de uma forma ordenada que se repete em toda a sua extensão. (SOUZA, 2013)

2.4.3 - Entradas e Saídas

A placa Arduino UNO possui pinos de entrada e saídas digitais, assim como pinos de entradas e saídas analógicas, em seguida é exibido a pinagem conhecida como o padrão Arduino:

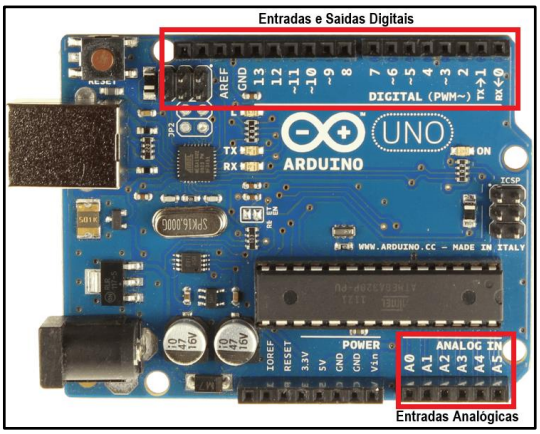

Figura 14 - Circuito de comunicação do Arduino. Fonte: O Autor

Conforme exibido na Figura 14, a placa Arduino UNO possui 14 pinos que podem ser usados como entrada ou saída digitais. Estes pinos operam numa tensão de 5V, onde cada pino pode fornecer ou receber uma corrente máxima de 40 mA. Cada pino possui resistor de *pull-up* (tensão para cima) interno que pode ser habilitado por software. Alguns desse pinos possuem funções especiais: (SOUZA, 2013)

- PWM: 3,5,6,9,10 e 11 podem ser usados como saídas PWM de 8 bits através da função analogWrite();
- Comunicação serial: 0 e 1 podem ser utilizados para comunicação serial. Deve-se observar que estes pinos são ligados ao microcontrolador responsável pela comunicação USB com o PC;
- Interrupção externa: 2 e 3. Estes pinos podem ser configurados para gerar uma interrupção externa, através da função attachInterrupt().

Para interface com o mundo analógico, a placa Arduino UNO possui 6 entradas, onde cada uma tem a resolução de 10 bits. Por padrão estão ligadas internamente a 5V, ou seja, quando a entrada estiver com 5V o valor da conversão analógica digital será 1023. O valor da referência pode ser mudado através do pino AREF. (SOUZA, 2013)

2.4.4 - Microcontrolador

Na Figura 15, podemos ver em destaque o microcontrolador ATMEGA328P, que é um computador em um único componente, um circuito integrado o qual contém um núcleo de

processador, memória e periféricos programáveis de entrada e saída. A memória de programação pode ser RAM, NOR flash ou PROM a qual, muitas vezes, é incluída no chip. Os microcontroladores são concebidos para aplicações embarcadas, em contraste com os microprocessadores utilizados em computadores pessoais ou outras aplicações de uso geral. É programado para tarefas específicas, diferentemente de um microprocessador de propósito geral. (MICROCHIP, 2021)

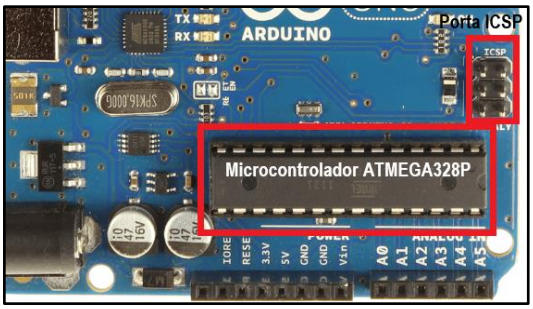

Figura 15 - Microcontrolador ATMEGA328P. Fonte: O Autor

A placa Arduino UNO é programada através da comunicação serial, pois o microcontrolador vem programado com o *bootloader*. Dessa forma não há a necessidade de um programador para fazer a gravação (ou upload) do binário na placa. A comunicação é feita através do protocolo STK500. A programação do microcontrolador também pode ser feita através do conector ICSP utilizando um programador ATMEL, em realce na Figura 15. (SOUZA, 2013)

2.5 - Sensor DHT11

O DHT11, na Figura 16, é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0 a 50 graus celsius e umidade entre 20 a 90%, muito usado para projetos com Arduino. Sua faixa de precisão pode variar até 2 graus de temperatura e 5% de umidade, este sensor possui 4 pinos, mas o terceiro pino não é utilizado. (THOMSEN, 2020)

Este sensor foi escolhido devido a sua facilidade de utilização e sua precisão nas medidas.

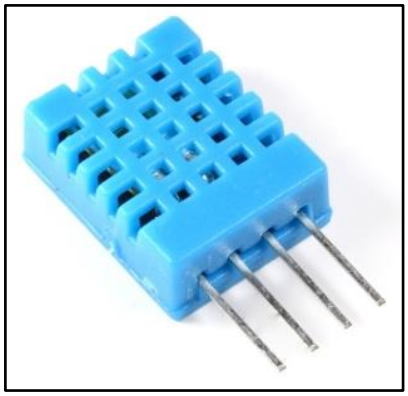

Figura 16 - Sensor de temperatura/umidade DHT11. Fonte: Modificado de Thomsen (2020, p. 2)

2.6 - Módulo Ethernet ENC28J60

O módulo é formado pela porta ethernet RJ45, o chip controlador ENC28J60 (daí o nome do módulo, datasheet), um cristal de quartzo de 25Mhz e um conector de 10 pinos, o que torna prática a ligação desse módulo à diferentes tipos de microcontrolador. A ligação deste módulo ao Arduino é feita via interface SPI (*Serial Peripheral Interface* - é um protocolo que permite a comunicação do microcontrolador com diversos outros componentes, formando uma rede). (MONK, 2014)

O ENC28J60, mostrado na Figura 17, foi o controlador de rede escolhido para permitir a comunicação dos componentes na rede. Este controlador de rede ethernet comunica-se a 10 Mbps, e possui memória de 8 Kb. (MONK, 2014)

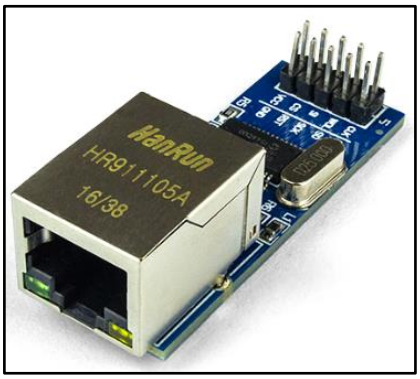

Figura 17 - Módulo Ethernet ENC28J60. Fonte: Modificado de Robocore.net (2020, p. 1)

Uma vantagem deste controlador sobre os demais é permitir a configuração do seu endereço físico e o uso do protocolo de comunicação é o SPI. Ele é uma especificação de interface de comunicação série síncrona usada para comunicação em curta distância, em sistemas embarcados e afins. A interface do SPI foi desenvolvida pela empresa Motorola e tornou-se, através do seu uso frequente, um padrão de fato. Os dispositivos SPI comunicam entre si em modo "full duplex" usando uma arquitetura "mestre-escravo" com um único mestre. O dispositivo mestre origina a comunicação para a leitura e a escrita. Múltiplos dispositivos

escravos são suportados através da utilização de linhas de seleção de escravos individuais. (MONK, 2014)

É utilizado para atribuir ao Arduino a conexão ethernet/internet, dessa forma torna-se possível controlar o Arduino a partir da rede interna (ethernet) ou através da rede externa (internet).

2.7 - JSON - *JavaScript Object Notation*

É um formato aberto de arquivo que consiste na formação de pares de atributos, valor e matrizes (pacote de dados). Sua principal vantagem é a facilidade de interpretação e leitura dos dados pelos programas que o recebem, no caso o Zabbix. É utilizado para transmissão de informações entre sistemas diversos. (DEVMEDIA, 2020)

JSON (*JavaScript Object Notation*) é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido a análise dessas informações. Isto explica o fato de o JSON ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados. (DEVMEDIA, 2020)

A ideia utilizada pelo JSON para representar informações é tremendamente simples: para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado. Esta sintaxe é derivada da forma utilizada pelo *JavaScript* para representar informações. Por exemplo, para representar uma identificação “1”, nome “Fulano” e uma nota “10”, utiliza-se a seguinte sintaxe: (DEVMEDIA, 2020)

Exemplo: ["id": 1, "nome": Fulano, "nota": 10]

Conforme exposto no exemplo acima, as variáveis id, nome e nota tem valores atribuídos como “1”, “Fulano” e “10”, respectivamente.

Em um outro exemplo, considerando a comunicação utilizada neste projeto:

["host":"arduino-sensor","key":"umid","value":"15.00"]

No exemplo exposto acima, são definidas as variáveis “host”, “key” e “value”. E, seus respectivos valores são: arduino-sensor, umid e 15.00.

Através do modo de transmissão *Unicast* (transmissão feita por um emissor, destinada a apenas um receptor na rede), o JSON foi a maneira mais simples e eficaz de acessar as informações geradas pelo sensor de temperatura e umidade e enviá-las para visualização no

monitoramento, por isso, será o formato que utilizaremos para trocar informações entre o Arduino e o Zabbix.

2.8 - Zabbix – Software de Monitoramento de Redes

O Zabbix é um software que monitora vários parâmetros da rede, dos servidores e da disponibilidade dos serviços. Utiliza-se de um mecanismo flexível de notificação que permite configurar alertas para praticamente qualquer evento. As notificações permitem que se reaja rapidamente a problemas no ambiente. O Zabbix oferece excelentes recursos de relatórios e visualização de dados armazenados. Isso faz com que o Zabbix seja a ferramenta ideal para planejamento operacional da rede de computadores. Foi criado por Alexei Vladishev, e atualmente é mantido e suportado pela Zabbix SIA. É uma solução robusta, de código aberto e com suporte a monitoração distribuída. (ZABBIX, 2020)

A arquitetura Zabbix e a flexibilidade dos módulos permitem que a ferramenta seja utilizada para o monitoramento convencional (ligado/desligado, on/off), acompanhamento de desempenho de aplicações, análise de experiência de usuário e análise de causa raiz em ambientes complexos, através do servidor Zabbix e suas regras de correlacionamento, a exemplo: verificar se um servidor físico está ativo, com conexão de rede, ou se um serviço web deste servidor está em pleno funcionamento. (ZABBIX, 2020)

Essa ferramenta 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. (ZABBIX, 2020)

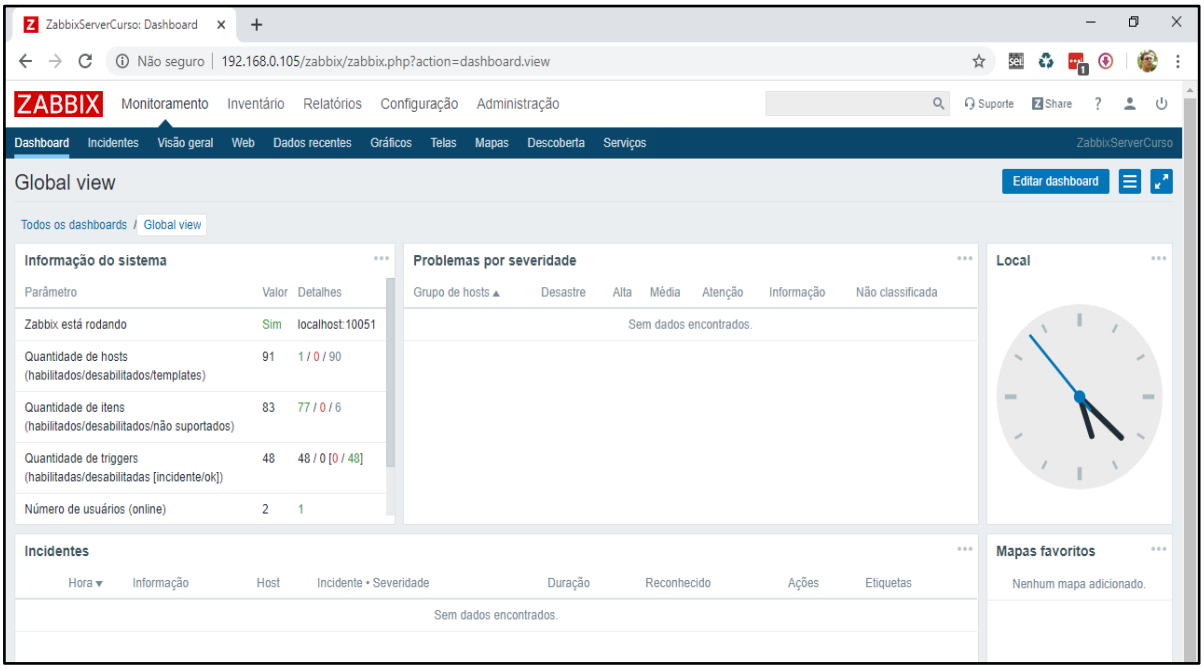

Figura 18 - Dashboard Zabbix 4.0. Fonte: O Autor

Na Figura 18, é mostrado o *Dashboard*, ou seja, a tela de monitoramento do Zabbix.

É com esse software que iremos receber os dados dos itens temperatura e umidade, gerando alertas, caso alguns desses dados fiquem fora das faixas estabelecidas nas *Triggers* (as triggers são expressões lógicas que analisam os dados coletados pelos itens e representam o estado do sistema em relação aos mesmos). (ZABBIX, 2020)

CAPÍTULO 3 – METODOLOGIA

A identificação do problema e solução escolhidos, deu-se de forma espontânea, no dia a dia no trabalho no HU-UFS (Hospital Universitário de Sergipe), observando-se os racks instalados nos diversos setores sem climatização adequadas, e os problemas ocasionados pelo não monitoramento das temperaturas e umidades, chegando a ter que parar os serviços de alguns desses equipamentos.

A metodologia deste trabalho foi baseada em uma pesquisa descritiva sobre o tema, onde objetiva-se descrever as características e funções de cada um componente utilizado para o projeto proposto. Dessa forma, procurou-se demonstrar como essas partes funcionam e como podem se integrar para implementar o produto final.

O trabalho desenvolvido utilizará como base:

- O Arduino UNO, que é formado por um circuito integrado Atmega328, ele possui 14 pinos digitais de entrada e saída, 6 entradas analógicas, um cristal de quartzo de 16 MHz, conexão USB, um conector de energia, um botão ICSP (*In Circuit Serial Programming*) e um botão de reset. (SOUZA, 2013)
- O sensor para obtenção dos valores de temperatura e umidade é o DHT11. Por conta do sensor de temperatura/umidade (lê temperaturas no intervalo de 0ºC a 50°C e umidade do ar no intervalo de 20% a 90%) que usaremos para este projeto, faremos uma média dos dois dados apresentados anteriormente: a recomendação da ASHRAE e a especificação do fabricante, resultando em limiares onde a temperatura poderá variar entre 9ºC a 36ºC e a umidade entre 23% e 70% que vai servir como base para definir as *Triggers* utilizadas nos itens temperatura e umidade no sistema Zabbix, para aplicação no monitoramento desses dados e posterior análise. É muito importante salientar, que os limiares que utilizaremos não ferem as recomendações do fabricante do equipamento de referência. (THOMSEN, 2020)
- Um módulo *Ethernet* ENC28J60 para permitir a comunicação dos componentes na rede, assim sendo possível controlar o Arduino a partir da rede interna (ethernet) ou através da rede externa (internet). Após a comunicação estabelecida e através do modo de transmissão *Unicast*, será utilizado o formato JSON para troca dos dados entre o Arduino e o Zabbix.

O projeto desenvolvido é para ser aplicado em racks simples de telecomunicação, localizados nos mais diversos ambientes e fora dos padrões recomendados, nas pequenas

empresas e instituições que não dispõe de muitos recursos financeiros para adquirir uma solução mais robusta. Também partindo da premissa que estes locais não têm nenhum tipo de climatização adequada, este projeto será ideal e de baixo custo, com eficiência para monitorar a temperatura e umidade desses racks, gerando alertas visuais, sonoros e envio de e-mails se assim desejar, caso os valores obtidos de mínima/máxima de temperatura e umidade ultrapassem a média dos dados limiares que são: temperatura entre 9ºC a 36ºC e a umidade entre 23% e 70%.

Após todas as etapas de desenvolvimento do projeto até a fase de testes, com a compra dos equipamentos necessários (*hardware*), a estimativa dos custos foi determinada por meio de pesquisas de preços de mercado, reduzindo o máximo possível os custos, para ser uma solução barata e eficaz. Através desse processo conseguimos ter uma base de valores que será utilizado para realizar o projeto na prática, que podem sofrer uma variação média de 10% para mais ou para menos. Portanto, o valor estimado dos custos do projeto para compra do *hardware* utilizado é de R$ 151,80 (cento e cinquenta e um reais, e oitenta centavos) podendo sofrer alterações dependendo da loja onde for comprado. No capítulo Resultados e Discussão analisaremos os custos com mais detalhes.

CAPÍTULO 4 – PROPOSTA E EXPERIMENTO

Neste capítulo vamos apresentar os procedimentos que foram utilizados para a solução do problema proposto neste Trabalho de Conclusão de Curso. O passo a passo de como se deu o desenvolvimento do projeto.

4.1 - Leitura de Temperatura e Umidade no Monitor Serial

O primeiro passo para a construção do projeto, foi estabelecer o escopo da aplicação, definindo os locais a serem monitorados, no caso, os racks de telecom. e os dados a serem acompanhados:

- **Temperatura** - Monitorar a temperatura do ambiente é importante porque os equipamentos instalados em um rack de telecom., nos mais variados ambientes, durante o seu funcionamento, geram muito calor e um valor elevado pode causar sobreaquecimento dos servidores e consequentemente desligamento ou até mesmo danos irreversíveis ao mesmo;
- **Umidade** - Alta umidade no ambiente que se encontra o rack, pode provocar condensação de água nos servidores, já a baixa umidade pode gerar curtos dentro dos servidores devido a geração de carga eletrostática.

Definido o primeiro passo, então fomos para fase de testes com o sensor DHT11. Montamos o circuito eletrônico entre o sensor e o Arduino. O DHT11 possui 4 terminais sendo que somente 3 são usados: GND, VCC e Dados, onde foram respectivamente ligados nas portas GND, 5V e porta digital 2 do Arduino, como mostrado na Figura 19.

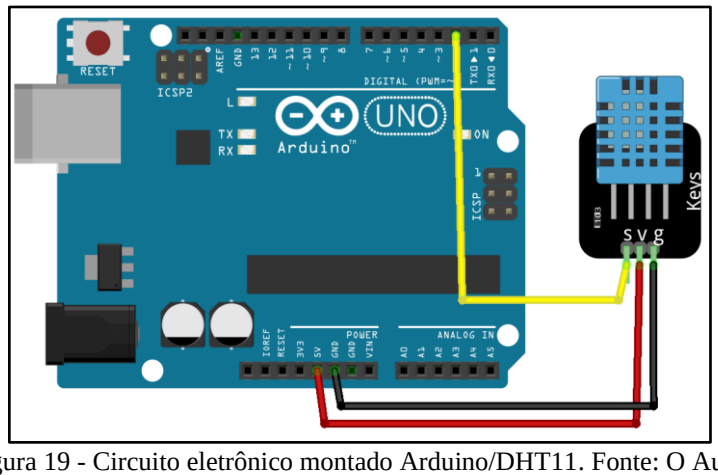

Figura 19 - Circuito eletrônico montado Arduino/DHT11. Fonte: O Autor

Já existe uma biblioteca chamada “DHT.h” disponível gratuitamente no *Github* (plataforma de hospedagem de código-fonte e arquivos com controle de versão) e podendo ser alterada conforme as necessidades. (INDUSTRIES, 2020)

Após várias tentativas, implementamos o código, chegamos em resultado satisfatório, onde na Figura 20 são mostrados o código e o monitor serial, com a saída dos dados de temperatura e umidade coletados pelo sensor. Fazendo uma breve tradução do código abaixo, basicamente o Arduino lê as informações do sensor, verifica se não tem nenhuma falha do sensor, caso não, as envia para serem mostradas no monitor serial.

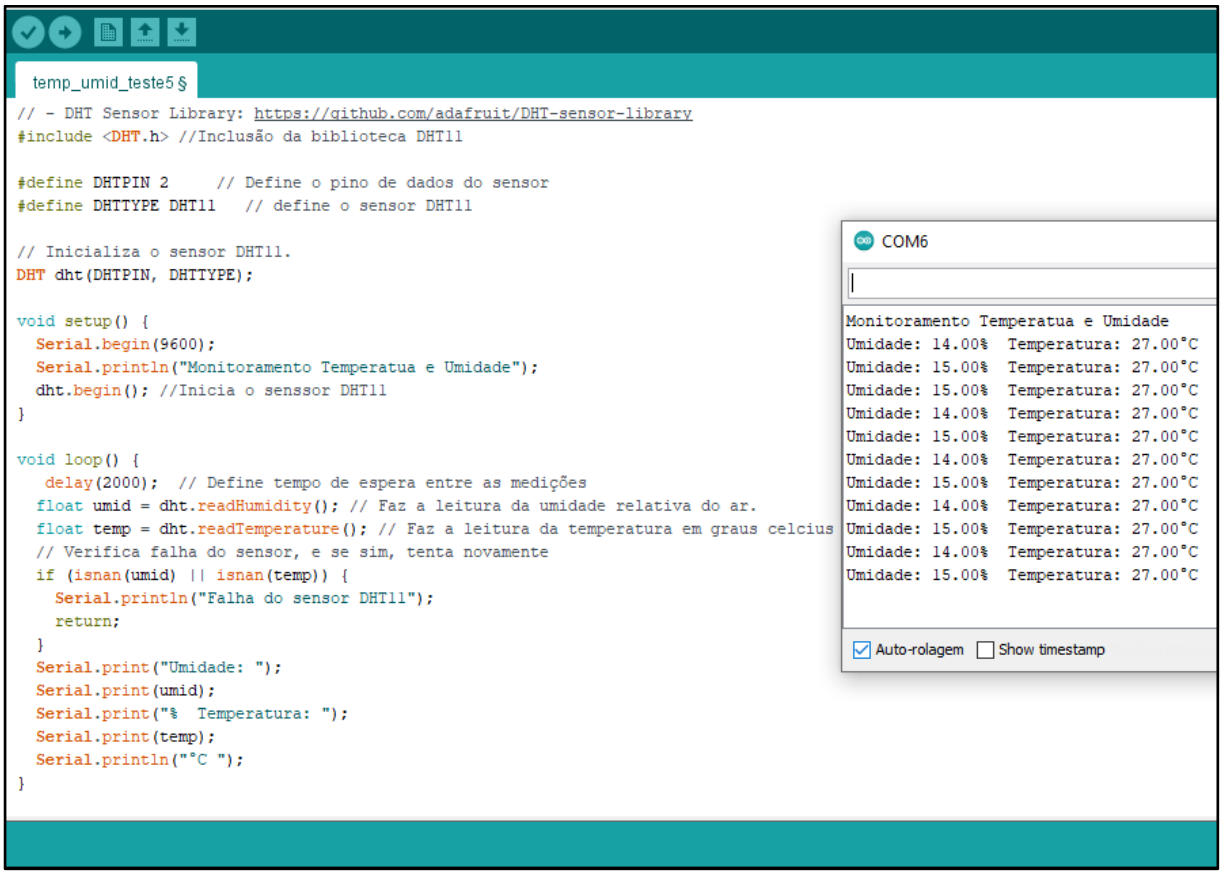

Figura 20 - Código e resultado da leitura dos dados do sensor DHT11. Fonte: O Autor

Como pode ser observado na Figura 20, depois dos testes na prática, percebemos que o sensor DHT11 não tem a precisão para fazer a leitura com as casas decimais, ou seja, ele não fornece informações “quebradas” de temperatura e umidade. Isso significa que o sensor vai mostrar as informações de, por exemplo, 25, 26, 27 graus, mas não as casas decimais de 25,7 ou 27,3 graus, igualmente para a umidade.

4.2 - Servidor *Web com* Módulo *Ethernet* ENC28J60

A proposta na prática desse teste, é utilizar o módulo *Ethernet* ENC28J60 em conjunto com o Arduino e o Sensor DHT11 para fazer uma conexão à um *webserver* e acessar uma página web que irá exibir as informações de temperatura e umidade.

Após a conclusão dos testes do sensor DHT11 terem sidos concluídos com sucesso, partimos para os testes do módulo *Ethernet* ENC28J60 juntamente com o sensor de

temperatura/umidade, onde montamos um servidor *web* para mostrar os dados coletados pelo sensor. O desenvolvimento desse teste foi feito em duas partes: primeiro a montagem dos componentes e segundo o desenvolvimento da aplicação responsável pelas leituras das informações e exibi-las em tempo real numa página *web*, criada na linguagem HTML, onde ela foi inserida dentro do escopo do código do Arduino. Ao inserir o Arduino na internet, você pode acessá-lo de qualquer local do mundo, seja com um computador, smartphone ou tablet e obter informações ou solicitar que ações sejam executadas.

Salientamos que esses testes individuais foram necessários, para se ter a noção de como cada item funciona separadamente conectado ao Arduino e como se comporta cada código implementado.

Como no teste anterior já estava montado o circuito eletrônico do sensor DHT11, só acrescentamos o *Ethernet Shield*, com a seguinte modificação: como o *Shield* utiliza a porta digital 2 do Arduino como padrão, trocamos a porta de dados do sensor para a porta 7 do Arduino, como pode ser mostrado nas Figuras 21 e 22, e Tabela 1.

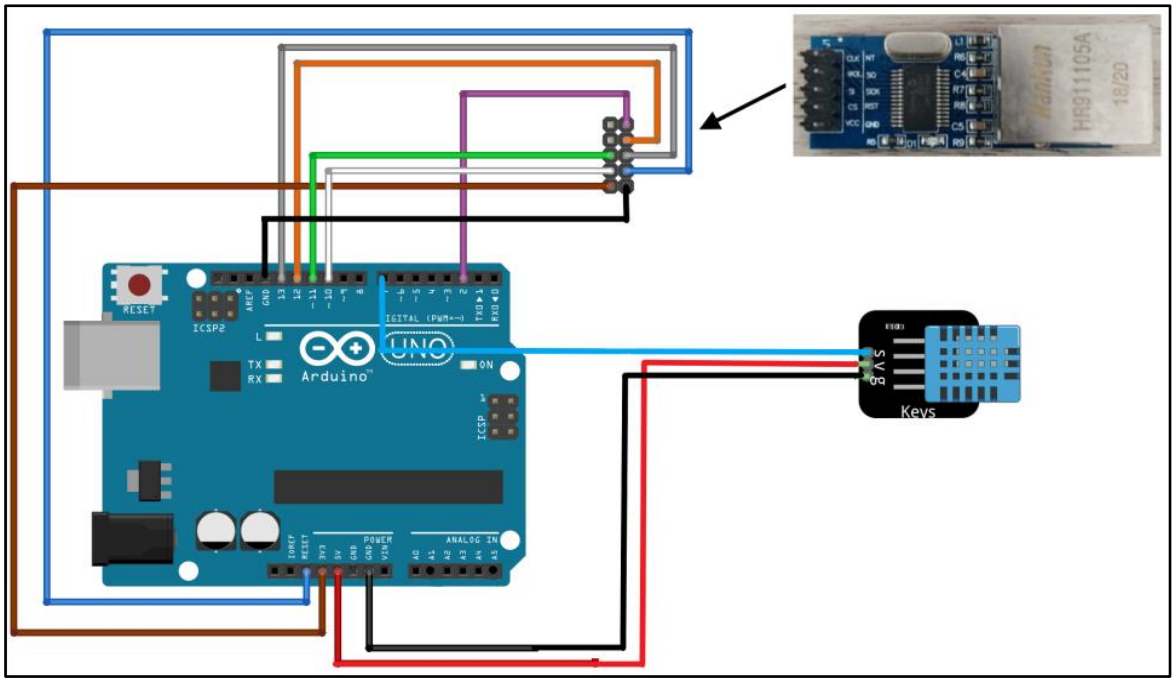

Figura 21 - Esquema circuito eletrônico para o servidor web. Fonte: O Autor

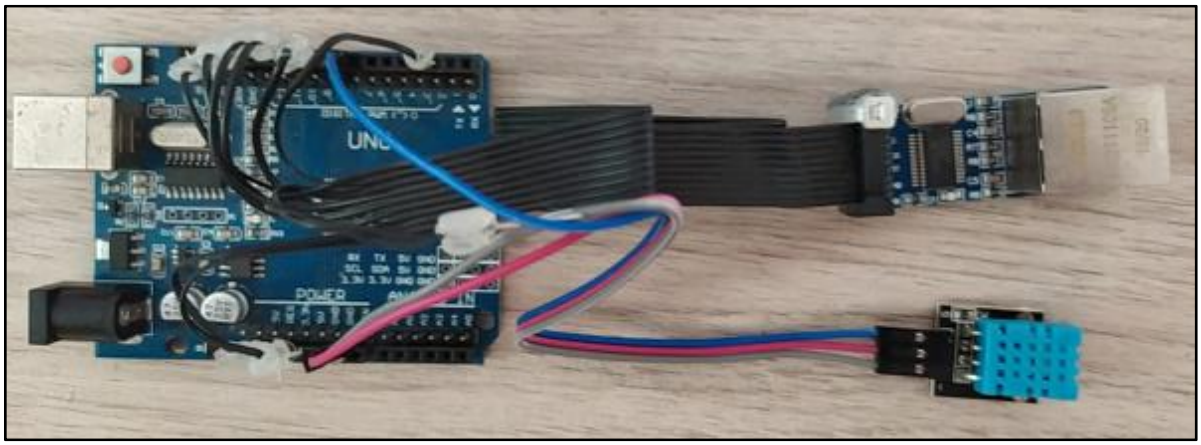

Figura 22 - Foto real do circuito montado. Fonte: O Autor

Tabela 1 - Conexão do ENC28J60 as Porta dos Arduino

Conector Módulo ENC28J690	Porta que será conectado no Arduino UNO
CS	10
SI	11
SO	12
SCK	13
RESET	RESET
NT	2
VCC	3.3V
GND	GND

Fonte: O Autor

Para o desenvolvimento do software no IDE do Arduino, usaremos como base o código inicial dos testes com o sensor DHT11, já deixo registrado, que esse código será usado até a fase final do projeto, só fazendo as devidas modificações e implementações necessárias.

No código foi incluída as bibliotecas “UIPEthernet.h” versão 2.0.6, baixada do *Github* (TRUCHSESS, 2020), de outras bibliotecas testadas, essa foi a que apresentou melhores resultados, e a “SPI.h” já é inclusa por padrão no IDE do Arduino.

Como pode ser observado na Figura 23 e mencionado anteriormente, a segunda parte do teste é a aplicação desenvolvida no IDE do Arduino. Podemos observar que o código foi dividido em duas partes, para melhor entendimento. Nessa primeira parte, foi incluída as bibliotecas necessárias, definidos os endereços MAC e IP para o *Ethernet Shield*, o tipo do sensor e as variáveis que receberão as leituras de temperatura e umidade usando como parâmetros a função da biblioteca, como também foi informada uma porta para o servidor de conexão *Web*. Logo após, foram passados os parâmetros para a função de conexão com a rede, e iniciado o sensor DHT11, e em seguida cria-se uma conexão com o cliente, no caso o navegador *Web*.

Verificar

webserver_dht_final §

```
//INCLUSÃO DAS BIBLIOTECAS
#include <SPI.h> //Biblioteca permite comunicação do Shield ENC28J60 c/ placa arduino
#include <UIPEthernet.h> //Biblioteca Shield ENC28J60
#include <DHT.h> //Inclusão da biblioteca DHT11

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xF5, 0xE8 }; //Atribui endereço MAC ao Shield ENC28J60
byte ip[] = { 192, 168, 0, 158 }; ////Atribui endereço IP ao Shield ENC28J60
EthernetServer server(80); //Porta de conexão do servidor WEB
#define DHTPIN 7     // Define o pino de dados do sensor
#define DHTTYPE DHT11   // define o tipo sensor DHT11
DHT dht(DHTPIN, DHTTYPE); // Inicializa o sensor DHT11.

void setup(){
  Ethernet.begin(mac, ip); //Inicia e atribui os parâmetros para função de conexão com a rede.
  server.begin(); //Inicia o servidor para receber dados na porta 80.
  dht.begin(); //Inicia sensor DHT11
  }

void loop(){
    delay(2000);  // Define tempo de espera entre as medições
    float temp = dht.readTemperature(); // Faz a leitura da temperatura em graus celcius
    float umid = dht.readHumidity(); // Faz a leitura da umidade relativa do ar.
  EthernetClient client = server.available(); //Cria uma conexão com o cliente
  if (client) { // Verifica se o cliente existe, e faz...
    while (client.connected()) {//Enquanto existir cliente conectado, faz
    if (client.available()) { //Verifica se o cliente está conectado, faz
    char c = client.read(); ////Define variável do cliente e lê para requisição HTTP
          if (c == '\n')
```

Figura 23 - Primeira parte do código do Servidor *Web*. Fonte: O Autor

Na segunda parte do código mostrado na Figura 24, foi implementada uma página *Web*, com o título “Servidor Arduino” em linguagem HTML, que fará a exibição das informações de temperatura e umidade em tempo real, após acessar por qualquer navegador (no exemplo foi usado o Edge) o endereço http://192.168.0.158, como visualizado na Figura 25.

```
//Código de criação da página Web (Liguagem HTTP)
        client.println("HTTP/1.1 200 OK"); //Versão do HTTP
        client.println("<!DOCTYPE html>");
        client.println();
        client.println("<html>");
          client.println("<head>");
            client.println("<meta charset='utf-8'/>");
            client.println("<title>Servidor Arduino</title>");
          client.println("</head>");
            client.println("<body style=background-color:#ADFF2F>"); //Define cor do fundo da página Web
            client.println("<center><font color='blue'><h1>TCC - TRABALHO DE CONCLUSÃO DE CURSO</font></center></h1>");
              client.println("<h1><center>MONITORAMENTO DE TEMPERATURA E UMIDADE - DHT11</center></h1>");
                  client.println("<br/>"); //Quebra linha
              client.println("<br/>");
              client.println("<br/>");
              client.println("<hr />");
              client.println("<center><font size='6'>Temperatura: ");
                  client.println(temp); //Informa o valor atual da temperatura no Servidor WEB
                  client.println(" °C");
                  client.println("</center>");
              client.println("<hr />");
              client.println("<center><font size='6'>Umidade: ");
                  client.println(umid); //Informa o valor atual da umidade no Servidor WEB
                  client.println("%");
                  client.println("</center>");
               client.println("<hr />");

        client.println("</body></html>"); //Finaliza a TAG "body" E "html"
        client.stop(); //Finaliza a requisição HTTP e desconecta o cliente.
         }
        }
      }
    }
```

Figura 24 - Continuação, segunda parte código do Servidor *Web*. Fonte: O Autor

Figura 25 - Servidor *Web* Arduino em funcionamento. Fonte: O Autor

4.3 - Monitoramento de Temperatura e Umidade no Zabbix com Arduino UNO

4.3.1 – Código IDE Arduíno

Concluídos os testes com sucesso dos principais componentes do projeto, agora vamos partir para implementação final, de fato o que o trabalho propõe, que é monitoramento de temperatura e umidade através do Zabbix. Usando o conhecimento adquirido nas fases anteriores, faremos uma junção, podemos assim dizer, dos dois códigos desenvolvidos,

realizando algumas modificações necessárias para o funcionamento correto do software, sendo que o cabeçalho do código é igual do Servidor *Web*, por isso não vamos expor a parte inicial, somente a parte que foi incluída, no caso, o grande diferencial é a implementação da função *Zabbix Sender*, que é a responsável por enviar os dados no formato JSON para o servidor Zabbix.

Após pesquisa extensa de como desenvolver a função *Zabbix Sender* no Arduino, e alguns testes fracassados por a biblioteca escolhida do *Ethernet Shield* não ser compatível com a base de códigos encontrados em alguns fóruns e sites, encontramos um canal no *Youtube* chamado *IoTools*, que nos deu uma referência significativa de como implementar essa função no Arduino. Adaptamos o código apresentado no canal, de acordo com a nossa necessidade, e com testes preliminares conseguimos enviar as informações do sensor DHT11, via *post* JSON para o servidor Zabbix, através da função *Zabbix Sender*. (SILVA, 2018)

A diferença no início do código, é que foram declaradas 03 (três) variáveis globais (temp, umid e falhaSensor) que são usadas pela função *Zabbix Sender*. E foram criados 03 (três) arquivos de códigos, para deixar mais organizado e fácil de atualizar. Sendo um principal que contém as funções Setup() e Loop(), e os outros dois as funções config_inicial() que é chamada na função Setup() e zabbix_sender() que é chamada na função Loop(), como mostrado na Figura 25.

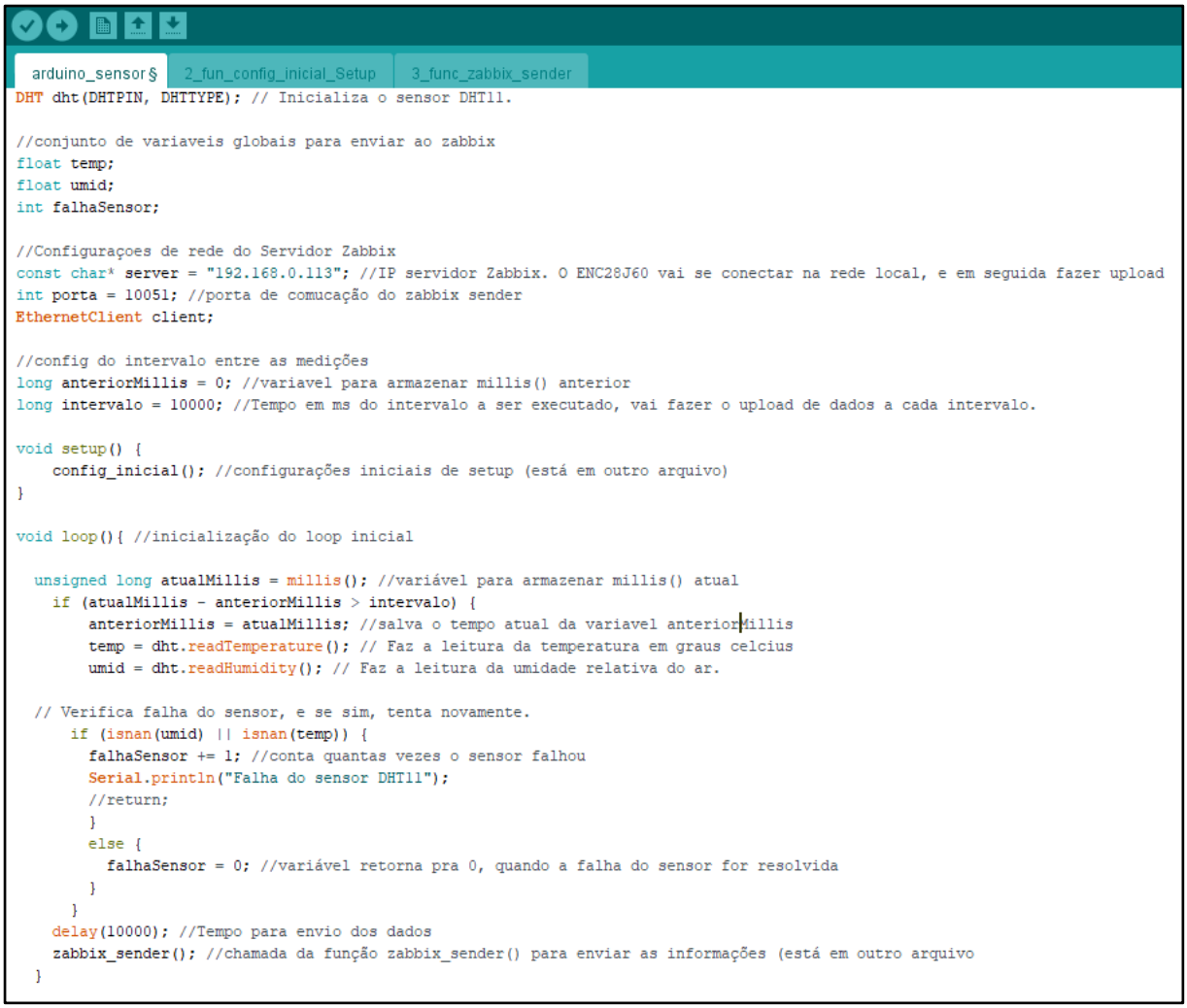

Figura 26 - Implementação inicial arquivo “arduino-sensor”. Fonte: O Autor

As linhas do código da Figura 26 estão comentadas para melhor entendimento, mas basicamente faz o seguinte:

1. cria as variáveis globais;
2. declara as variáveis “server” que recebe o IP do servidor Zabbix e “porta” onde definimos a porta de comunicação do Zabbix Sender;
3. inicia a conexão com o cliente (servidor Zabbix);
4. declara as variáveis “anteriorMillis” e “intervalo”, a primeira armazena o tempo em milissegundos da função millis(), a segunda o tempo do intervalo para envio dos dado do sensor DHT11;
5. dependendo da condição (*if*) executada no momento, em seguida chama a função config_inicial() apresentada na Figura 27.
6. Verifica se há falha no sensor, caso sim, incrementa 1 na variável “falhaSensor” e retorna ao início do *loop(),*
7. caso não, chama a função zabbix_sender() e faz a leitura dos dados do sensor.

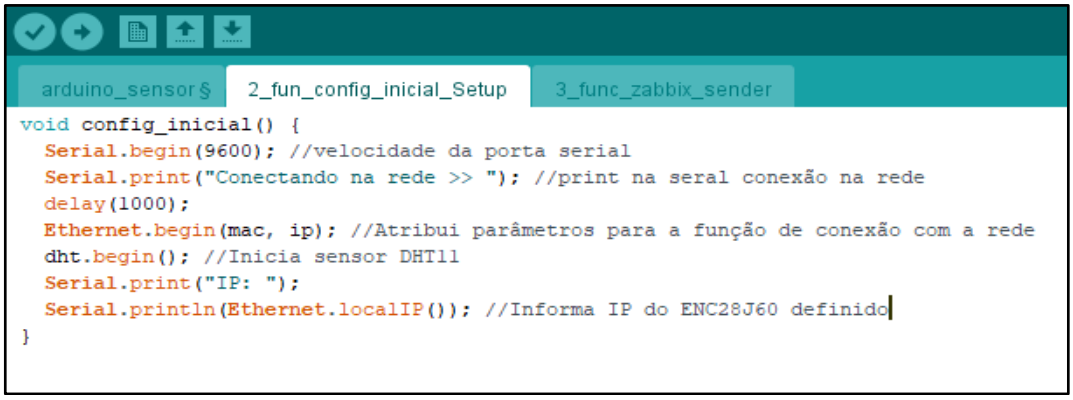

Figura 27 - Arquivo função config_inical(). Fonte: O Autor

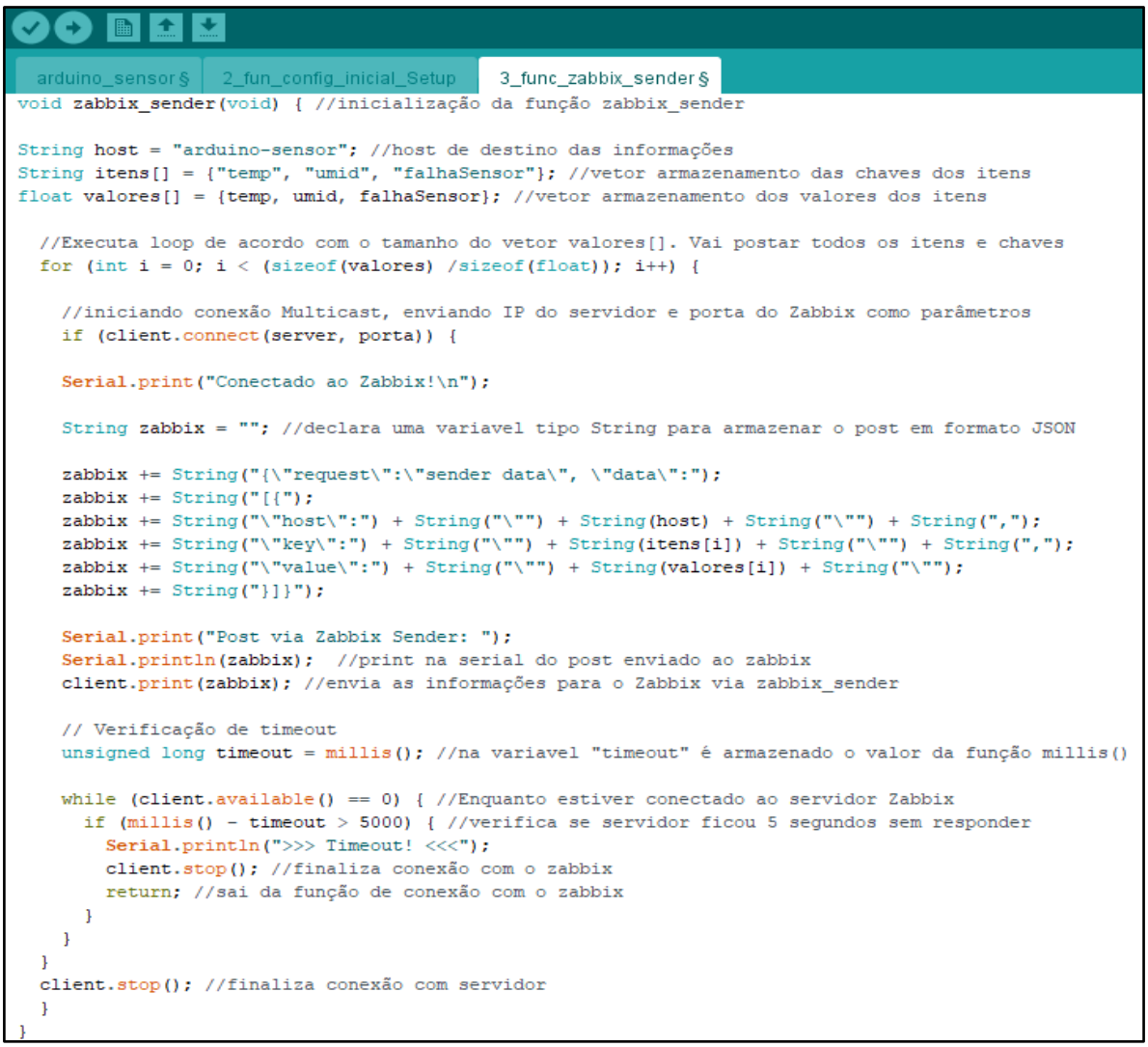

Figura 28 - Arquivo função zabbix_sender(). Fonte: O Autor

Comentando a Figura 28:

1. inicialmente na função zabbix_sender() é declarada a variável “host” que vai receber o nome do host que será criado no servidor Zabbix;

2. cria-se os vetores *itens*[] e *valores*[] que receberão os nomes das chaves e os valores das variáveis globais respectivamente. Para que os dados sejam enviados com sucesso, é necessário que o nome do host e itens(chaves) criados no servidor Zabbix sejam idênticos aos declarados na função zabbix_sender().
4. verifica a conexão com o cliente e passa os parâmetros “server” e “porta”,
5. e declara a variável “zabbix”, após, monta o post JSON e armazena na variável “zabbix”.
6. verifica a condição de conexão do cliente.
7. envia para o servidor Zabbix conectado e finaliza a conexão com o servidor.

Na Figura 29, é mostrado o resultado do teste de envio dos dados do sensor DHT11, como também uma simulação de “Falha do Sensor” para o Servidor Zabbix. E ainda é mostrado os dados através do monitor serial no momento do envio. Vejam que os dados são iguais.

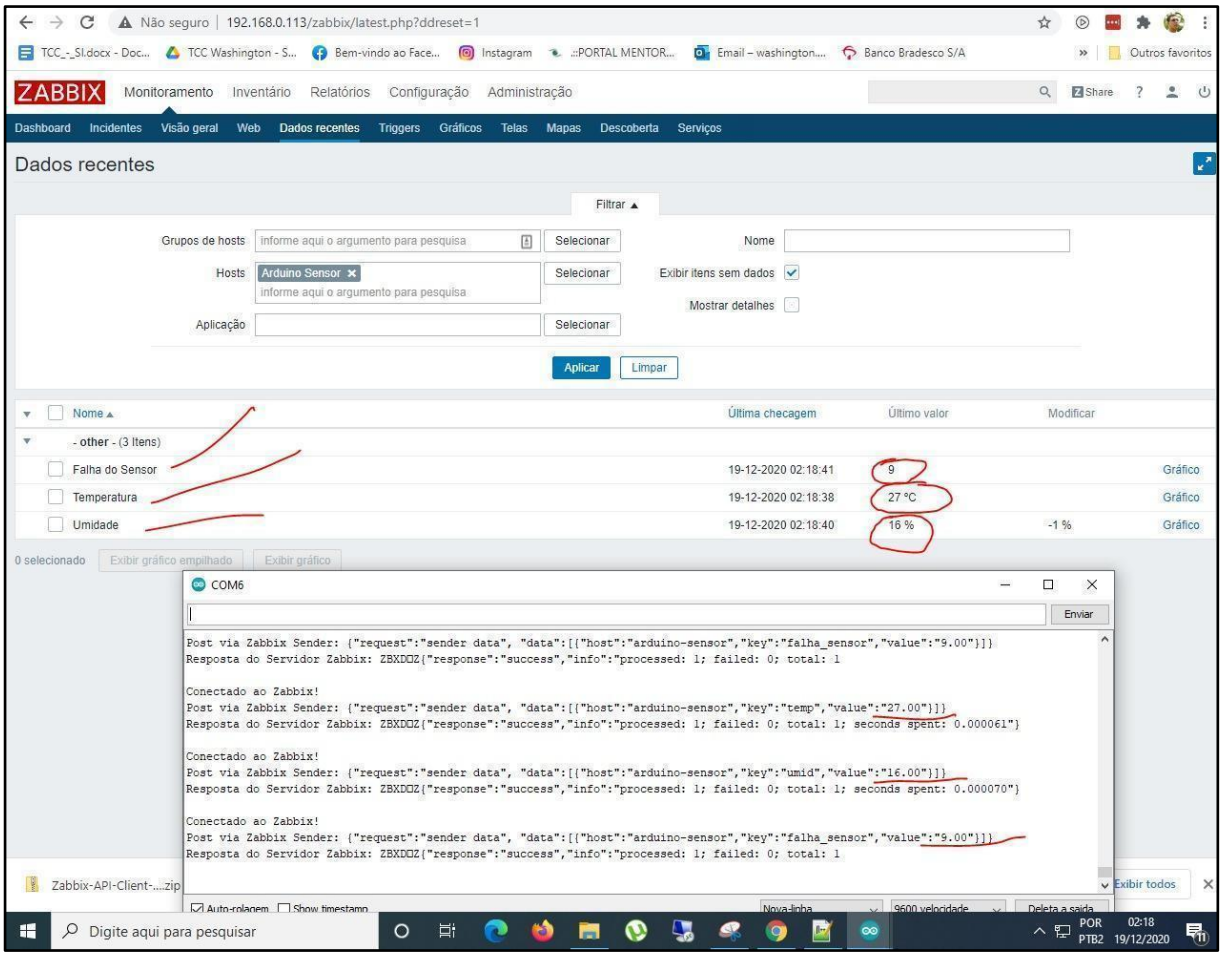

Figura 29 - Demonstração de envio dos dados para o Servidor Zabbix. Fonte: O Autor

4.3.2 – Instalação, configuração e gerenciamento do Servidor Zabbix

Para a instalação e configuração do servidor Zabbix, foi utilizado o *VitualBox 6.1* (VIRTUALBOX, 2020), onde foram instaladas duas máquinas virtuais com o Sistema Operacional *Debian Server 64x* (DEBIAN, 2020) e instalações do Zabbix versões, 3.4 e 4.0 (ZABBIX, 2020). A máquina virtual que está em execução, está rodando o Zabbix 3.4 a qual foi usada para os respectivos testes em laboratório, como mostra a Figura 30.

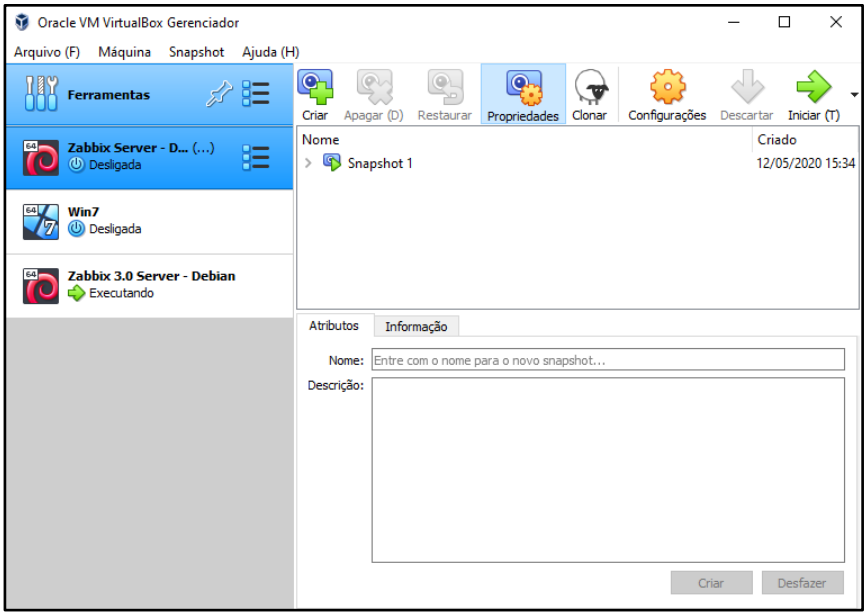

Figura 30 - Máquinas virtuais. Fonte: O Autor

O primeiro passo para início do monitoramento, foi a criação do grupo “Arduino” e do *host* “Arduino Sensor”, através do menu configuração/hosts. Nas Figura 31 é mostrada a tela de criação de *hosts.*

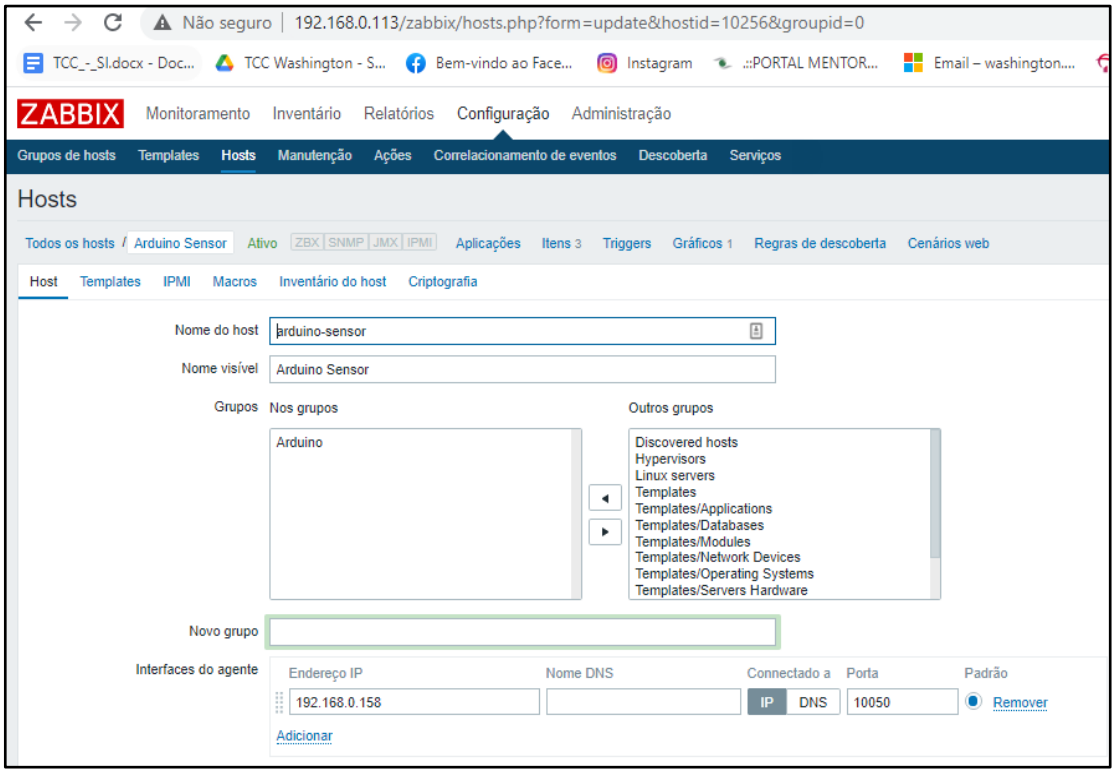

Figura 31 - Tela de criação do host. Fonte: O Autor

O “nome do *host*” como dito anteriormente, tem que ser igual ao declarado na função zabbix_sender(), o “nome visível” fica a critério, já o novo grupo criado foi “Arduino”, e colocamos o IP do host, que no caso é o IP definido para o módulo *Ethernet* ENC28J60 e porta padrão do Zabbix, nesse caso outras configurações não foram necessárias. Após conferidas as informações só clicar em adicionar, e o host será criado.

Com o “host” já criado, agora vamos criar os “itens” a serem monitorados: Temperatura, Umidade e Falha do Sensor, como mostrado na Figura 32. Damos o nome pra o “item”, escolhemos o tipo “*Zabbix Trapper*” (a monitoração feita através do *Zabbix Trapper* aceitam dados enviados ao invés de ir buscá-los (ZABBIX, 2020)), e definimos o nome da “chave” que também tem que ser igual ao declarado na função zabbix_sender(), o tipo da informação recebida e definimos a unidades. Após conferir clicar em adicionar. Usamos o comando “clone para criar os demais “itens”. Na Figura 33, podemos observar os “itens” já criados.

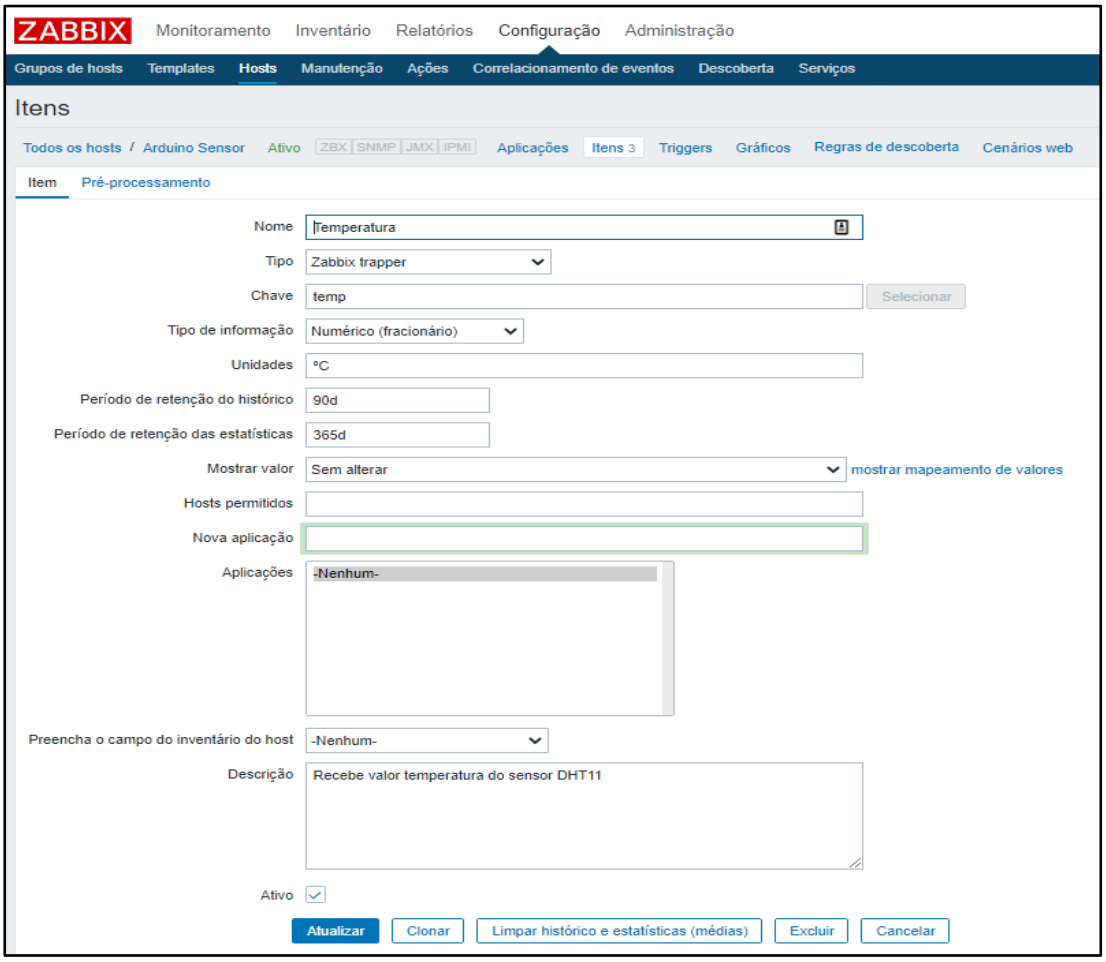

Figura 32 - Tela de criação do item. Fonte: O Autor

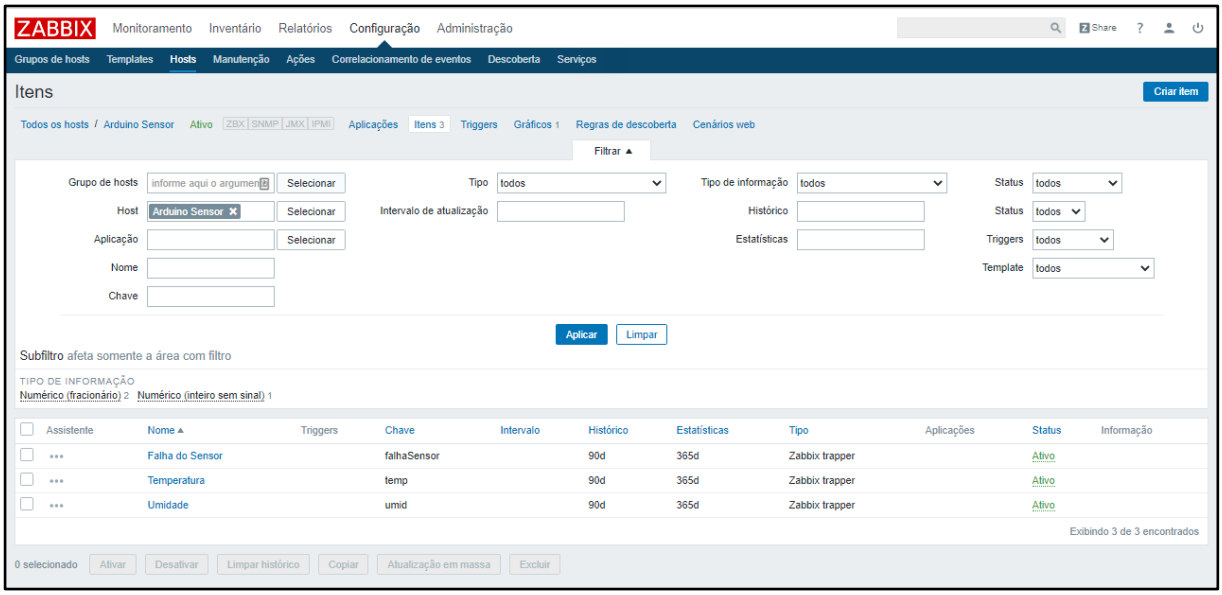

Figura 33 - Itens criados para o host *Arduino Sensor*. Fonte: O Autor

Para criação das *Triggers* para geração dos alertas, vamos nos basear nos limiares médios definidos anteriormente, onde a temperatura poderá variar entre 9ºC a 36ºC. De forma

básica e simples, damos um nome a *Trigger*, selecionamos a severidade e construímos expressão que desejamos, caso ela seja verdadeira ou falsa, nesse caso se a temperatura ficar abaixo de 9ºC ou acima 36ºC será gerado um alerta com a severidade média na tela do Zabbix. Veja na Figura 34 a tela de criação de *Triggers.*

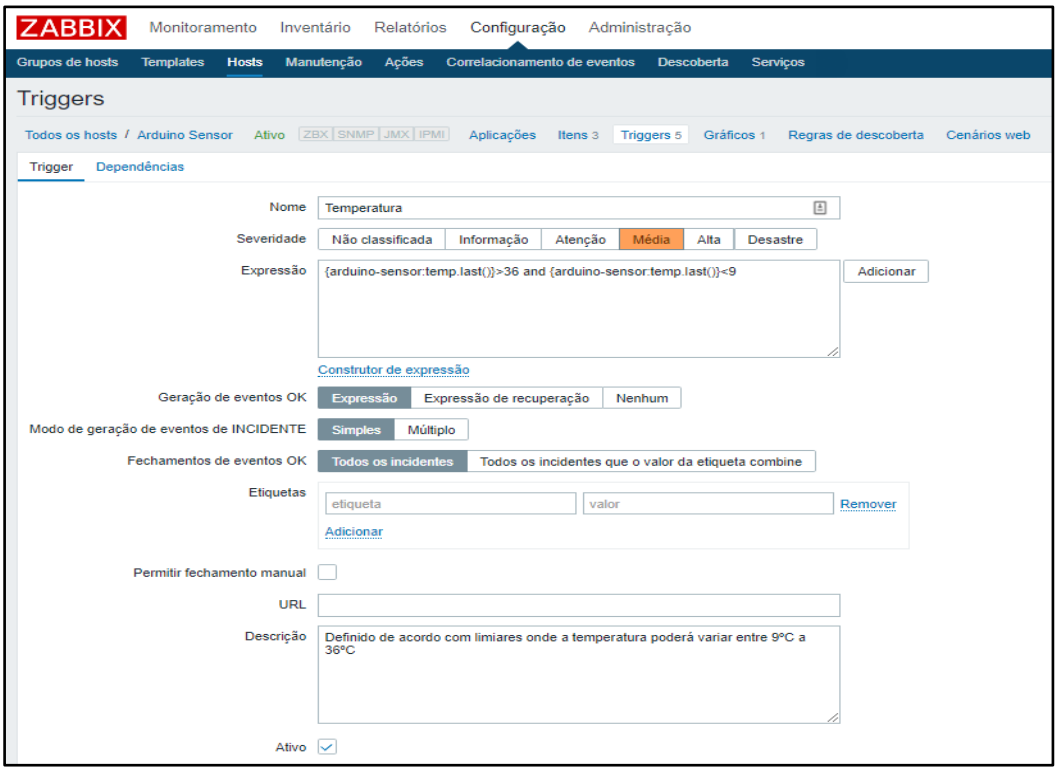

Figura 34 - Tela de criação de *Triggers*. Fonte: O Autor

Terminado os testes em laboratório, podemos destacar que os resultados foram satisfatórios, de acordo com a proposta apresentada no trabalho.

4.4 – Protótipo final do “Arduino Sensor”

Após conclusão das etapas de desenvolvimento dos softwares (criação do código no IDE do Arduino, instalação e configuração do Zabbix) com os resultados dos testes mostrados anteriormente, vamos agora para montagem do *hardware*, ou seja, o protótipo final do “Arduino Sensor”, que assim o batizamos, para que seja instalado facilmente em um ambiente de produção, onde será usado na prática, que no próximo capítulo apresentaremos os resultados.

Para montagem do *hardware* (Arduino UNO, sensor DHT11 e módulo *Ethernet* ENC28J60), que será nosso protótipo, reaproveitamos uma caixa metálica de um *drive* leitor de cartão de memória, onde adaptamos de acordo com as necessidades, para que os periféricos fossem instalados da melhor forma possível, como é mostrado nas Figuras 35 a 39.

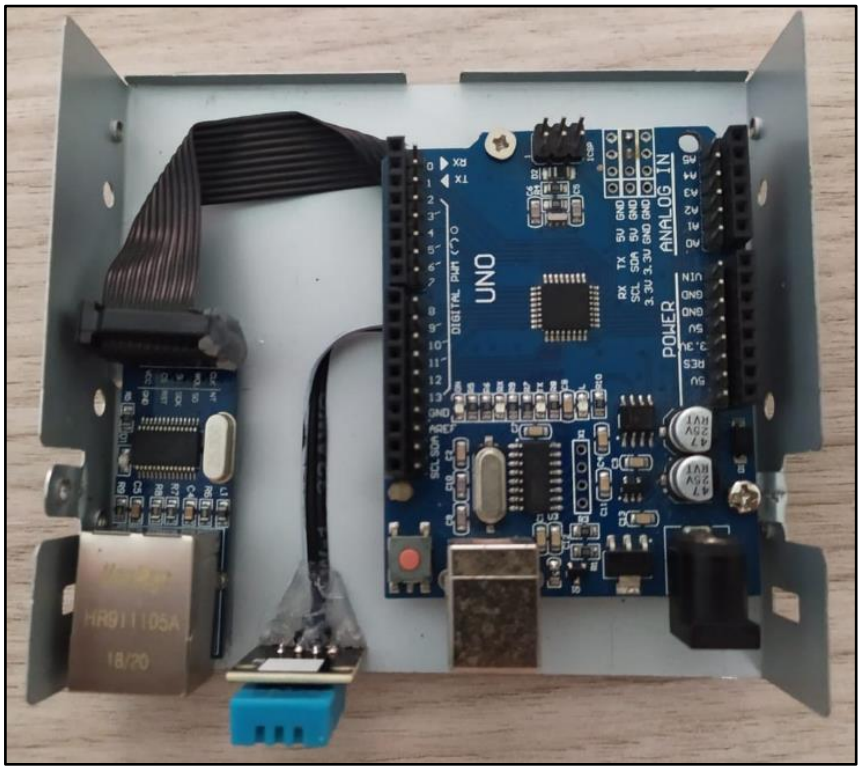

Figura 35 - Periféricos instalados na caixa metálica (visualização superior). Fonte: O Autor

Vale ressaltar que para conexões do sensor DHT11 e *Shield Ethernet* reaproveitamos fitas *flats* de conexão HD/IDE, onde adaptamos e soldamos direto na placa do Arduino UNO, para que não causasse perda de contato eletrônico entre eles. Assim dando uma melhor experiência no uso em racks de telecom.

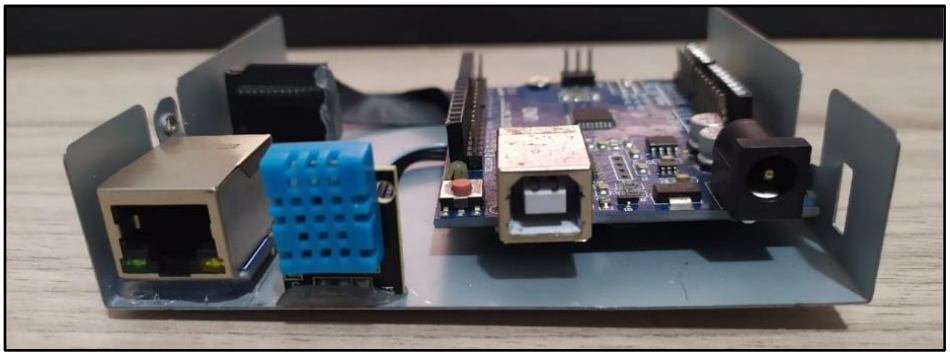

Figura 36 - Periféricos instalados na caixa metálica (visualização frente, caixa aberta). Fonte: O Autor

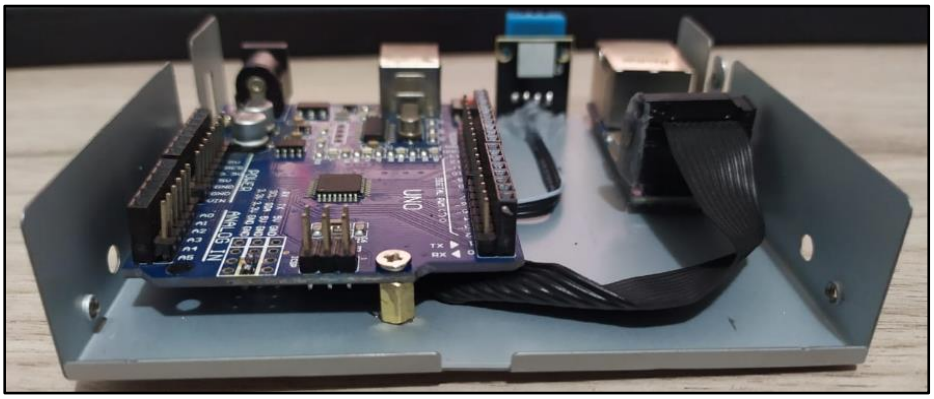

Figura 37 - Periféricos instalados na caixa metálica (visualização traseira, caixa aberta). Fonte: O Autor

Figura 38 - Periféricos instalados na caixa metálica (visualização frente, caixa fechada). Fonte: O Autor

Figura 39 - Periféricos instalados na caixa metálica (visualização traseira, caixa fechada). Fonte: O Autor

CAPÍTULO 5 – RESULTADOS E DISCUSSÃO

5.1 – Análise dos custos

O *hardware* utilizado no projeto, pode ser encontrado para compra em diversos sites especializados em eletrônica e robótica, como por exemplo a "Robocore" e "FilipeFlop", as duas lojas mais famosas e confiáveis encontradas no ramo. Na Tabela 2, podemos ver o valor detalhado de cada periférico usado, que foi orçado no site "www.robocore.net", assim conseguimos o melhor preço, fazendo com que a proposta do trabalho se concretizasse, que é uma solução de baixo custo.

Tabela 2 - Custos do projeto

ITEM	VALOR
Arduino UNO	R$ 59,90
Fonte 9V 1A Alimentação	R$ 16,50
Módulo *Ethernet* ENC28J60	R$ 29,00
Sensor DHT11	R$ 12,50
Frete	R$ 33,90
TOTAL	**R$ 151,80**

Fonte: (ROBOCORE.NET, 2020)

Como observamos detalhadamente na Tabela 2, com o valor total de R$ 151,80 (cento e cinquenta e um reais e oitenta centavos), conseguimos realizar o projeto, com os resultados esperados e relativamente barato, comparado a soluções prontas existentes no mercado. Vale reforçar que esse valor pode sofrer alteração para mais ou menos.

5.2 Análise da eficácia

Com o protótipo "Arduino Sensor" terminado e funcional como esperado, então, apresentaremos aqui os resultados obtidos na prática em um ambiente de produção. Para isso, pedimos autorização e ajuda do Analista de Redes do SGPTI (Setor de Gestão de Processos e Tecnologia da Informação) do HUPAA/UFAL (Hospital Universitário Prof. Alberto Antunes da Universidade Federal de Alagoas), que no momento estava como Chefe do setor, e autorizou, como também nos auxiliou na configuração no servidor Zabbix do HU (que atualmente está rodando a versão 4.0.2), na criação dos principais itens a serem monitorados, que são a temperatura e a umidade. A escolha do HUPAA para a obtenção dos resultados atingidos, foi de forma espontânea, pois já trabalho no mesmo como Técnico em Informática desde dezembro de 2017, portanto a acesso aos setores onde estão os racks é facilitado.

Antes da instalação física do projeto, alteramos o código do "Arduino Sensor" colocando as informações da rede e do servidor Zabbix do Hospital Universitário, e em seguida compilamos e carregamos no *hardware* do protótipo.

Foi disponibilizado para o teste em produção, o rack 29, localizado em uma sala climatizada, porém que não funciona em tempo integral, e está localizada no 5º andar (sala de aula 538). Neste rack, chegam todos os pontos de conexão de rede desse andar e a fibra óptica de conexão ao CDC, e estão instalados os seguintes equipamentos:

- 01 DIO
- 01 *Patch Panel* 24 Portas
- 01 Switch Huawei S5720 48 Portas

A instalação física do “Arduino Sensor” se deu de forma simples, rápida e fácil, não tendo nenhuma dificuldade, como era esperado. Veja na Figura 40.

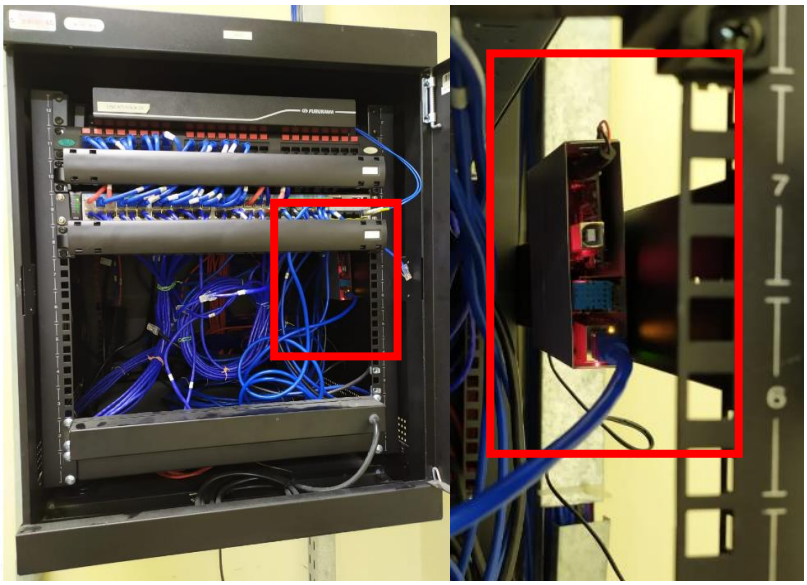

Figura 40 - Protótipo instalado no rack 29 no HUPAA. Fonte: O Autor

A criação dos itens e *triggers* no servidor Zabbix, foi feita de forma objetiva e sem problemas, sendo criados os principais itens de monitoramento, temperatura e umidade, e acrescentado mais dois itens que também são importantes, conexão de rede e falha do sensor, como pode ser visto nas Figuras 41 e 42.

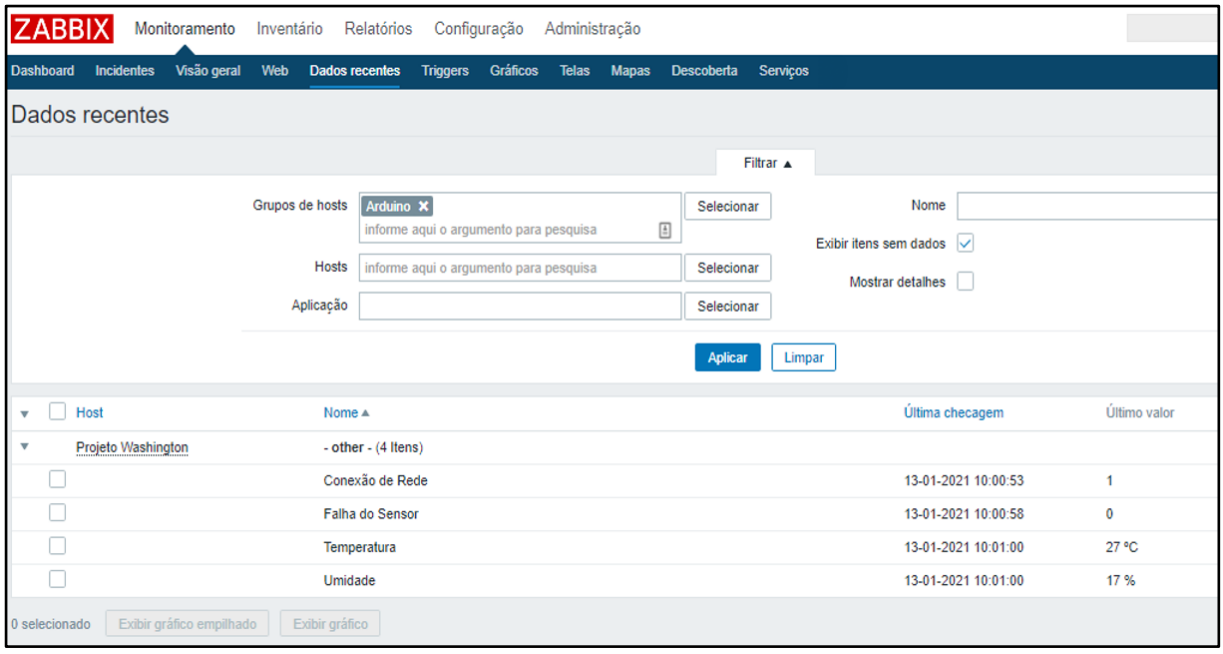

Figura 41 - Itens criado no servidor Zabbix do HUPAA. Fonte: O Autor

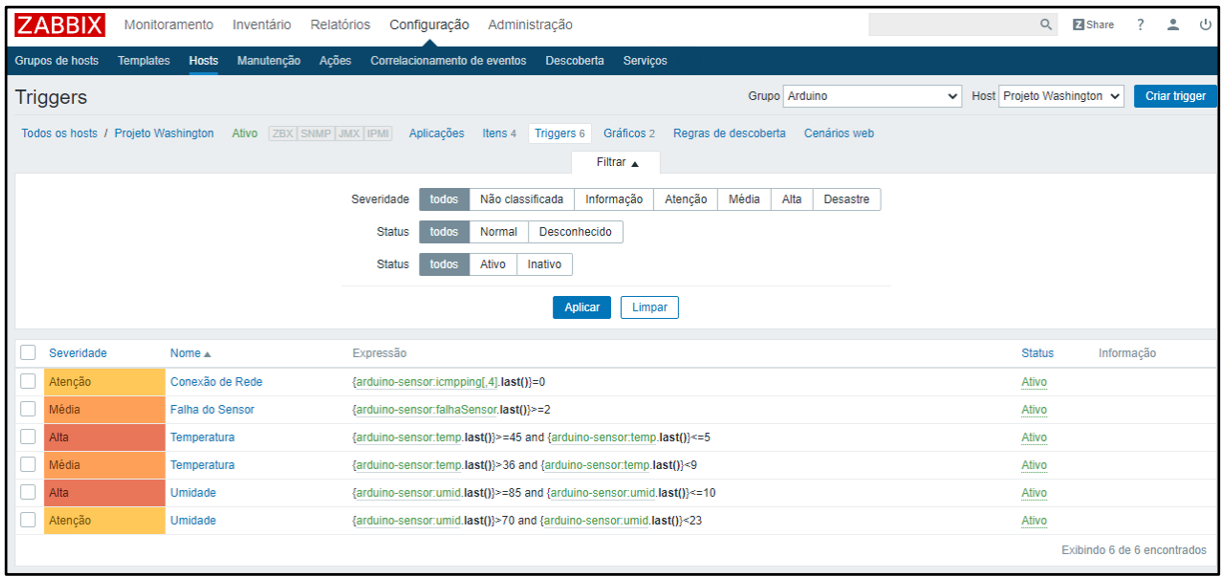

Figura 42 - *Triggers* criadas no servidor Zabbix do HUPAA. Fonte: O Autor

As expressões utilizadas nas *Triggers* para gerar os alertas de incidentes, mostradas na Figura 41, “dizem” basicamente o seguinte: Caso a temperatura for menor que 9 ou maior que 36 graus celsius, o Zabbix fará um alerta de severidade MÉDIA, mas se esses dados continuarem alterando, e fiquem menor/igual 5, ou maior/igual a 45 graus, o Zabbix fará um alerta de severidade ALTA. Mesma coisa acontece com o item umidade, se for menor que 23% ou maior que 70%, fará um alerta de severidade ATENÇÃO, mas se esses dados continuarem modificando, e fiquem menor/igual 10%, ou maior/igual a 85%, o Zabbix fará um alerta de severidade ALTA. Já no item “Falha no Sensor”, caso falhe ele contará quantas vezes o sensor falhou e registrará, e fará um alerta de severidade MÉDIA. No item conexão de rede, ele irá

fazer um "*ping*" para o "Arduino Sensor" frequentemente, verificando se há conexão estabelecida entre o protótipo e a rede de computadores, caso não, fará um alerta com severidade de ATENÇÃO.

Tivemos um problema no momento de recebermos os dados do "Arduino Sensor", nos deparamos com um erro de "*Timeout*", ou seja, o protótipo não estava conseguindo enviar as informações através da função zabbix_sender() e nem receber uma resposta do servidor Zabbix. Após uma análise detalhada do código e da documentação do Zabbix, percebemos que o erro estava na versão do servidor Zabbix, pois os testes em laboratório foram feitos na versão 3.4 e versão que está rodando no HU é a 4.0.2, então verificamos que o formato do *post* JSON a ser enviado, muda em relação a versão utilizada na fase de testes. Desta forma, tivemos que voltar para a etapa de desenvolvimento, e fazer a modificação necessária na função zabbix_sender(), alterando o *post* JSON de acordo com a documentação do Zabbix 4.0 (ZABBIX, 2020), assim corrigindo o problema.

Para não comprometer o funcionamento do monitoramento de redes do HUPAA, foi criado um usuário de testes no servidor Zabbix, com restrições, assim podemos configurar, alterar e visualizar somente os *hosts*, itens e *triggers* utilizados no projeto. Na Figura 43 segue imagem do *Dashboard* em funcionamento no Hospital Universitário.

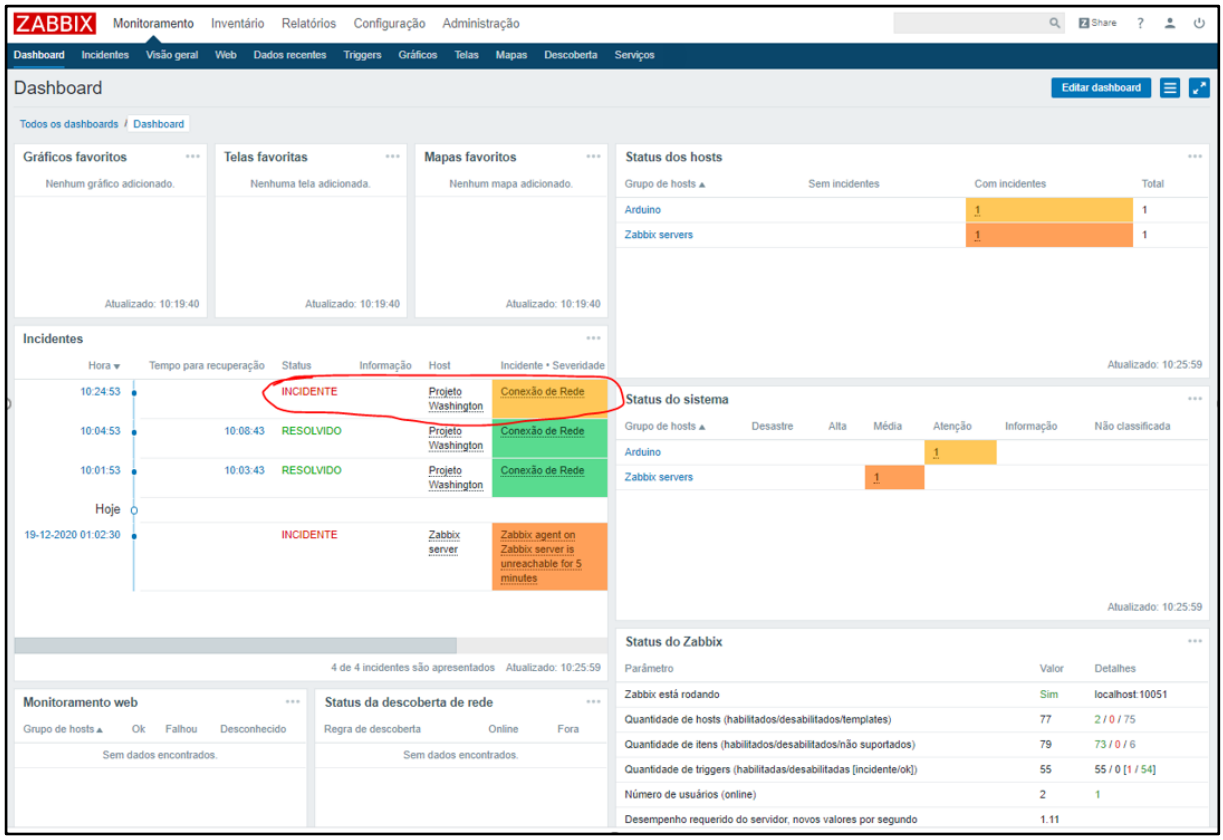

Figura 43 - *Dashboard* Zabbix no HUPAA. Fonte: O Autor

Para verificação da eficiência do projeto no ambiente de produção, o único teste controlado que é possível ser feito, é o de conexão de rede, onde desconectamos o cabo de rede do “Arduino Sensor” por 03 (três) vezes, e o Zabbix fez o ALERTA DE INCIDENTE configurado na respectiva *trigger* com sucesso, como era esperado, e pode ser verificado na Figura 42.

Já para os itens principais do projeto, temperatura e umidade, segue abaixo os gráficos produzidos no dia 13/01/2021, filtrado por 1 (uma) hora.

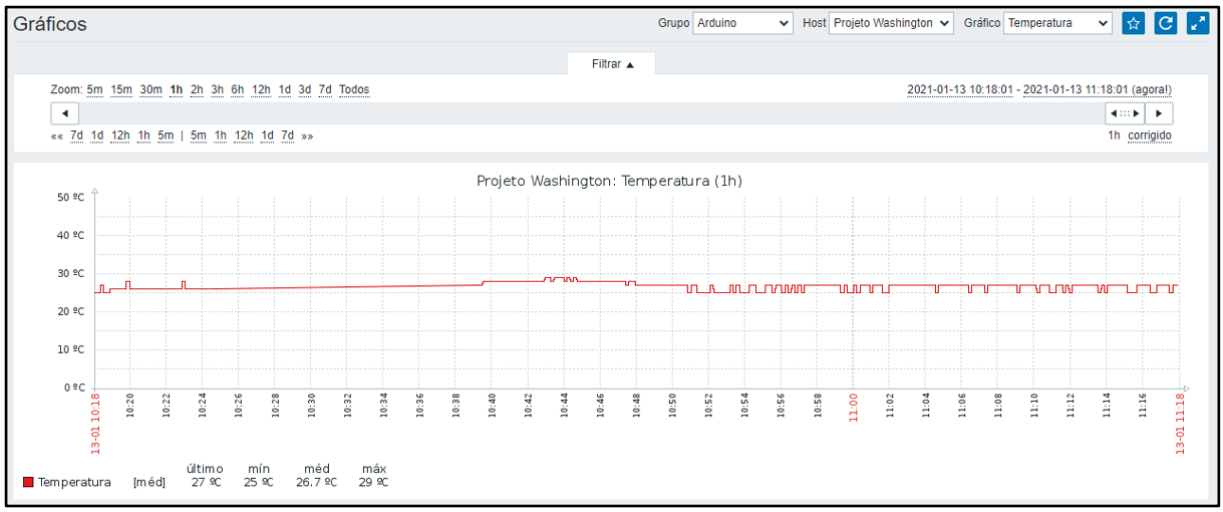

Figura 44 - Gráfico temperatura (escala de acordo com os limiares médios). Fonte: O Autor

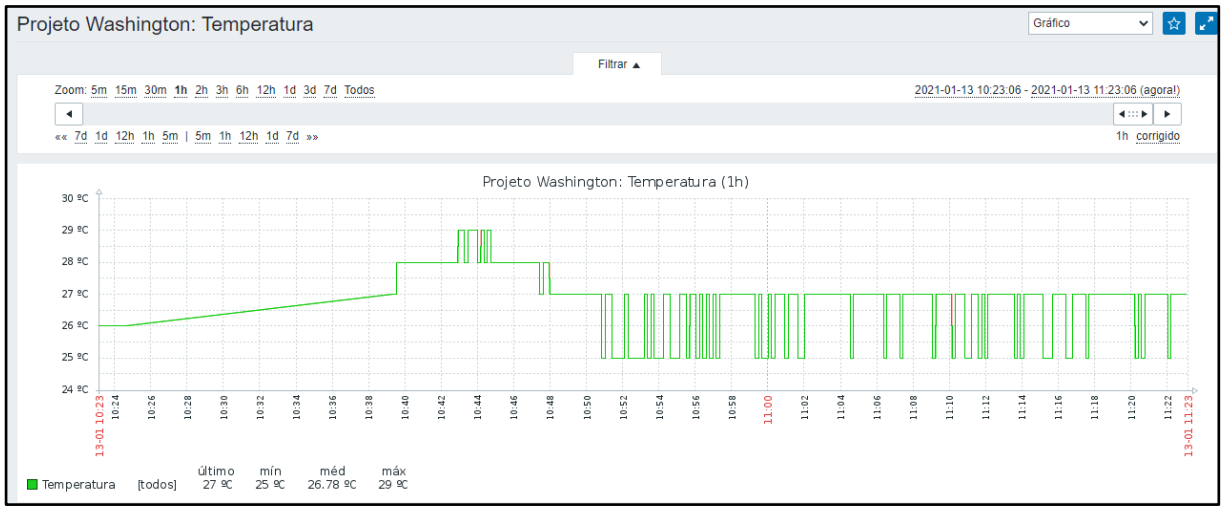

Figura 45 - Gráfico temperatura (escala com mínima e máxima dos valores obtidos). Fonte: O Autor

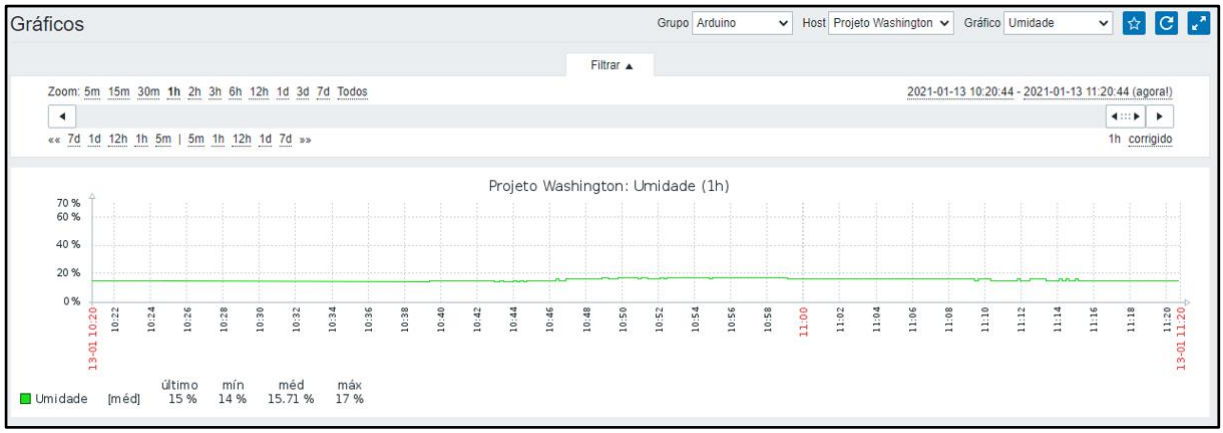

Figura 46 - Gráfico umidade (escala de acordo com os limiares médios). Fonte: O Autor

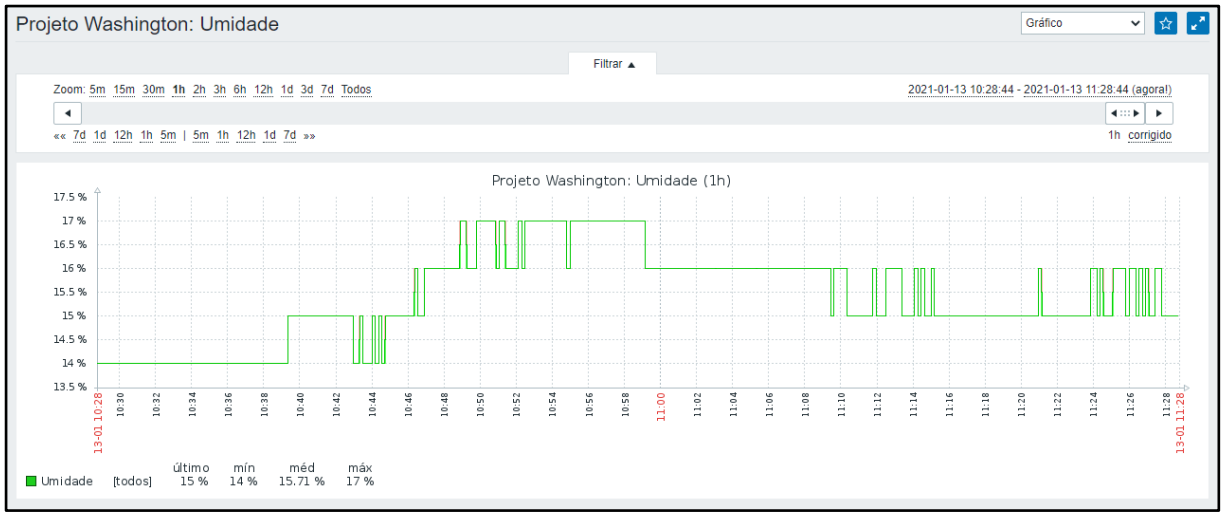

Figura 47 - Gráfico umidade (escala com mínima e máxima dos valores obtidos). Fonte: O Autor

Os resultados foram obtidos no ambiente de produção durante 03 (dias), de 11/01 à 13/01/2021, e a leitura dos gráficos acima é de um período de 01 (uma) hora, das 10:20h às 11:20h do dia 13/01/2021. Os dados foram aceitáveis, ratificando o objetivo do projeto, como pode ser verificado nas representações e *Dashboard* vistos anteriormente.

Nas Figuras 44 e 45, podemos perceber nos gráficos uma oscilação brusca e rápida da temperatura entre 25 e 27 graus em determinado período, isso aconteceu principalmente por dois fatores:

1- A faixa de precisão do sensor DHT11, que foi mencionado anteriormente no Capítulo 2, ela pode variar até 2 graus de temperatura e 5% de umidade;
2- Na fase de testes, percebemos que o sensor não faz leitura com casas decimais, ou seja, ele não fornece um valor real e sim um número inteiro.

Vale observar ainda que a sala de aula que está o rack, é equipada com um ar-condicionado, que é ligado ou desligado conforme a utilização da mesma pelos professores e alunos, e ocorre com bastante frequência durante o expediente, isso para justificar a leitura no começo do gráfico na Figura 45.

Diante do exposto acima, informamos que os dados estão corretos, e que isso não prejudica de forma alguma o propósito do projeto. Porém, o que ocorre é que os dados oscilam frequentemente dentro da faixa de precisão do sensor, que é de 2 graus. Como também não consegue elevar ou abaixar a temperatura gradativamente, porque não faz leituras com números decimais. O mesmo acontece com a umidade, conforme Figuras 46 e 47.

CAPÍTULO 6 – CONSIDERAÇÕES FINAIS

6.1 - Conclusões

O desenvolvimento na área de gerenciamento de redes, traz com ela novas tecnologias que podem ser aplicadas nos mais diversos fins. Um exemplo de aplicabilidade em racks simples de telecom, pois devido ao grau de importância, já que neles estão instaladas as "veias" e o "coração" que conectam a rede de computadores e outros periféricos das pequenas e grandes empresas ao mundo externo, por isso é fundamental o seu monitoramento.

O presente trabalho apresentou o "Arduino Sensor", uma ferramenta de baixo custo, composta por hardware e software para o **Monitoramento de Temperatura e Umidade em Racks Simples de Telecomunicação com Plataforma Arduíno.** Onde descrevemos o desenvolvimento do protótipo: Para se chegar ao objetivo, foi montado um circuito eletrônico entre um Arduino UNO, um sensor DHT11 e um *Ethernet Shield,* juntamente com um algoritmo para enviar as informações necessárias para o servidor Zabbix, responsável por receber os dados dos sensores e realizar o monitoramento dos principais itens, temperatura e umidade.

O desenvolvimento do protótipo serviu como prova de conceito da adoção de uma tecnologia de monitoramento de ambiente. A solução desenvolvida permite que o monitoramento possa ser realizado a distância permitindo a equipe de suporte monitorar as condições ambientais dos racks em horários de plantão, por exemplo.

A principal característica positiva do projeto é a facilidade de instalação física do *hardware* no rack de telecom. Já um ponto negativo, é que o projeto ficou com sua funcionalidade dependente do *software* Zabbix.

Portanto, é o momento de responder a pergunta de pesquisa formulada no capítulo 1: Sim, é possível construir uma solução de baixo custo para monitorar temperatura e umidade em racks simples de telecom, pois acabamos de ver e com resultados satisfatórios, o protótipo "Arduino Sensor" em operação num ambiente de produção no HUPAA/UFAL. Com isto não é difícil perceber que o Arduino é uma ferramenta em potencial a ser explorada, sendo de baixo custo e código aberto, o protótipo apresentado no controle de temperatura e umidade foi capaz de demonstrar a evolução de sua utilização, facilitando o monitoramento da temperatura e umidade no dia a dia.

6.2 - Sugestões para Trabalhos Futuros

A partir da experiência no desenvolvimento do protótipo, sugere-se como extensões ao presente trabalho as seguintes possibilidades:

a) Substituir o sensor DHT11 pelo DHT22, a fim de aumentar a capacidade de precisão e eficiência na leitura das variáveis do ambiente, assim conseguindo fazer leituras de números decimais;
b) Ou manter o DHT11, e antes de enviar os dados, fazer a leitura do sensor por três vezes, gerando uma média móvel dos valores obtidos;
c) Inclusão de um sensor para controle de abertura da porta do rack de telecomunicação.
d) Inclusão de um sensor de tensão AC – 0 a 250V, para controlar a energia que os equipamentos instalados no rack estão recebendo;
e) Implementação de uma MIB/SNMP para Arduino, onde seria disponibilizadas as informações do sensor, e o Zabbix ou outro software monitoramento iria buscar ao invés do “Arduino Sensor” enviá-las, assim deixando o projeto independente da utilização ou não do Zabbix;
f) Desenvolver um serviço *web*, disponibilizando os dados de temperatura e umidade com a plataforma Arduino, onde o Zabbix ou outro software de monitoramento pudesse consultar esses os valores obtidos pelo sensor.

REFERÊNCIAS

ALECRIM, E. Diferenças entre roteador, switch, modem e hub. **Info Wester**, 2019. Disponivel em: <https://www.infowester.com/hubswitchrouter.php>. Acesso em: 08 jun 2020.

ARDUINO. O que é Arduino? **Arduino**, 05 fev 2018. Disponivel em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 08 jun 2020.

CASE, J. et al. RFC 1157. **IETF Tools**, 1990. Disponivel em: <https://tools.ietf.org/html/rfc1157>. Acesso em: 02 fev 2021.

CISCO. Switch Catalyst 2960-S - Guia de primeiros passos. **Cisco**, 2012. Disponivel em: <https://www.cisco.com/c/dam/en/us/td/docs/switches/lan/catalyst2960/hardware/quick/guide_stack/All_languages/2960S_gsg_ptb.pdf>. Acesso em: 15 jun 2020.

CISCO. Switch Cisco Catalyst 2960S-48TS-L. **Cisco**, 2020. Disponivel em: <https://www.cisco.com/c/pt_br/support/switches/catalyst-2960s-48ts-l-switch/model.html>. Acesso em: 18 dez 2020.

COMPRASNET. Pregão Eletrônico. **ComprasNet**, 2018. Disponivel em: <http://comprasnet.gov.br/ConsultaLicitacoes/download/download_editais_detalhe.asp?coduasg=120623&modprp=5&numprp=82018>. Acesso em: 08 jun 2020.

COSENTINO, D. A importância da temperatura dos equipamentos de TI. **GD Solutions**, 2015. Disponivel em: <https://gdsolutions.com.br/monitoramento-de-ti/a-importancia-da-temperatura-dos-equipamentos-de-ti/#:~:text=Todos%20os%20servidores%20independente%20do,desempenho%20e%20aproveitamento%20da%20energia.&text=A%20temperatura%20ideal%20para%20uma,com%20varia%C>. Acesso em: 08 jun 2020.

DEBIAN. Baixando as imagens do CD/DVD Debian via HTTP/FTP. **Debian**, 2020. Disponivel em: <https://www.debian.org/CD/http-ftp/>. Acesso em: 02 fev 2020.

DEVMEDIA. Uma introdução ao JSON. **DevMedia**, 2020. Disponivel em: <https://www.devmedia.com.br/json-tutorial/25275>. Acesso em: 15 dez 2020.

FIGUEIREDO, Ê. Qual a temperatura ideal de um data center? **Redes Tecnologia e Serviços**, 2017. Disponivel em: <https://redestecnologia.com.br/qual-a-temperatura-ideal-de-um-data-center/#:~:text=Apesar%20de%20haver%20grande%20discuss%C3%A3o,27%C2%B0%20C%20com%20umidade>. Acesso em: 16 jul 2020.

HW GROUP. Poseidon2 3266. **HW Group**, 2021. Disponivel em: <https://www.hw-group.com/device/poseidon2-3266>. Acesso em: 02 fev 2021.

INDUSTRIES, A. DHT-sensor-library. **Github**, 2020. Disponivel em: <https://github.com/adafruit/DHT-sensor-library>. Acesso em: 22 dez 2020.

KUROSE, J. F.; ROSS, K. W. **Rede de Computadores e a Internet:** uma abordagem top-down. 5ª. ed. São Paulo: Pearson Education, 2010.

LAYERS COMMERCE. Rack Piso Servidor 19 20U P670 CWB METAL. **AZNET TELECOM**, 2020. Disponivel em: <https://www.aznettelecom.com.br/rack-acessorios/rack-piso/rack-piso-servidor-19-20u-p670-cwb>. Acesso em: 08 jun 2020.

LOTIERZO, R. **Alagamento do rack de telecom - Ambulatório geral HU-UFS**. Hospital Universitário da Universidade Federal de Sergipe. Aracaju-SE. 2016.

MARIN, P. S. **DATA CENTERS DESVENDANDO CADA PASSO:** Conceitos, Projeto, Infraestrutura Física e Eficiência Energética. [S.l.]: Érica, 2011.

MCROBERTS, M. **Arduino Básico**. 1ª. ed. São Paulo: Novatec, 2011.

MICROCHIP. ATmega328P. **Microchip**, 2021. Disponivel em: <https://www.microchip.com/wwwproducts/en/ATmega328P>. Acesso em: 02 fev 2021.

MKX E-COMMERCE. Rack 19" Para Parede 8U x 470mm Com Porta Frontal Acrílica. **UpperSeg**, 2020. Disponivel em: <https://www.upperseg.com.br/cftv/rack-organizador/rack-horizontal/rack-19-8u-x-450mm-para-parede-preto/>. Acesso em: 08 jun 2020.

MONK, S. **Projetos com Arduino e Android:** Use seu Smartphone ou Tablet para Controlar o Arduino. Porto Alegre: Bookman, 2014.

NATALMAKERS. Dispositivo: Conhecendo as partes do Arduino Uno. **Natalmakers**, 2015. Disponivel em: <http://natalmakers.blogspot.com/2015/08/dispositivo-conhecendo-as-partes-do.html>. Acesso em: 08 jun 2020.

QUADROS, D. O que são shields para Arduino? **FilipeFlop**, 2021. Disponivel em: <https://www.filipeflop.com/blog/o-que-sao-shields-para-arduino/>. Acesso em: 01 fev 2021.

ROBOCORE.NET. Kit Iniciante V8 Arduino - 2. O que é Arduino? **Robocore.net**, 2020. Disponivel em: <https://www.robocore.net/tutorials/kit-iniciante-v8-o-que-e-arduino>. Acesso em: 08 jun 2020.

SILVA, G. L. D. IOTOOLS. **Youtube**, 07 jun 2018. Disponivel em: <https://www.youtube.com/watch?v=2p1BoY2KGu0&list=FL6cb-Ja7AUIwxW5XGcg0CzA>. Acesso em: 20 dez 2020. 1 vídeo (18 min).

SOUZA, F. Arduino UNO. **Embarcados**, 29 nov 2013. Disponivel em: <https://www.embarcados.com.br/arduino-uno/>. Acesso em: 14 jul 2020.

THOMSEN, A. Monitorando Temperatura e Umidade com o sensor DHT11. **FILIPEFLOP**, 2020. Disponivel em: <https://www.filipeflop.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/>. Acesso em: 25 mai 2020.

TINKERCAD. **Autodesk Tinkercad**, 2020. Disponivel em: <https://www.tinkercad.com/dashboard>. Acesso em: 08 jun 2020.

TM TELECOM. Definião de Racks. **TM Telecom**, 2020. Disponivel em: <https://tmtelecom.com.br/produto/racks/>. Acesso em: 08 jun 2020.

TP-LINK. Switches Não Gerenciáveis - TL-SG1016D. **TP-Link**, 2020. Disponivel em: <https://www.tp-link.com/br/business-networking/unmanaged-switch/tl-sg1016d/>. Acesso em: 20 dez 2020.

TRUCHSESS, N. UIPEthernet. **Github**, 2020. Disponivel em: <https://github.com/UIPEthernet/UIPEthernet>. Acesso em: 22 dez 2020.

VIRTUALBOX. Download VirtualBox. **VirtualBox**, 2020. Disponivel em: <https://www.virtualbox.org/wiki/Downloads>. Acesso em: 04 ago 2020.

WBXRACKS. O que é U (Unidade de Rack)? **WBX Racks**, 2018. Disponivel em: <https://wbxracks.com.br/o-que-e-u-unidade-de-rack/>. Acesso em: 08 mai 2020.

ZABBIX. o que é o Zabbix? **Zabbix**, 2020. Disponivel em: <https://www.zabbix.com/documentation/4.0/pt/manual/introduction/about>. Acesso em: 14 jul 2020.

ZABBIX. Realize o Download e instale o Zabbix. **Zabbix**, 2020. Disponivel em: <https://www.zabbix.com/br/download>. Acesso em: 22 jul 2020.

ZABBIX. Zabbix Trapper. **Zabbix**, 2020. Disponivel em: <https://www.zabbix.com/documentation/4.0/manual/config/items/itemtypes/trapper>. Acesso em: 22 dez 2020.