UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS
BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

FELIPE CARDOSO PIMENTA

SISTEMA DE MONITORAMENTO PARA SALA DE SERVIDORES

PAU DOS FERROS - RN
2020

FELIPE CARDOSO PIMENTA

SISTEMA DE MONITORAMENTO PARA SALA DE SERVIDORES

Trabalho de Conclusão de Curso apresentado à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Tecnologia da Informação.

Orientador: Prof. Msc. Pedro Thiago Valério De Souza.

Co-orientador: Prof. Dr. Lenardo Chaves e Silva.

PAU DOS FERROS

2020

644s Pimenta, Felipe Cardoso.
SISTEMA DE MONITORAMENTO PARA SALA DE SERVIDORES / Felipe Cardoso Pimenta. - 2020.
57 f. : il.

Orientador: Pedro Thiago Valério De Souza.
Coorientador: Lenardo Chaves e Silva.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2020.

1. Arduino. 2. Sistema de Medição. 3. Interface Gráfica. 4. Banco de Dados. 5. Monitoramento. I. Souza, Pedro Thiago Valério De, orient. II. Silva, Lenardo Chaves e, co-orient. III. Título.

FELIPE CARDOSO PIMENTA

SISTEMA DE MONITORAMENTO PARA SALA DE SERVIDORES

Trabalho de Conclusão de Curso apresentado à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Tecnologia da Informação.

Defendida em: 07 / 02 / 2020.

BANCA EXAMINADORA

Pedro Thiago Valério de Souza

Pedro Thiago Valério De Souza, Prof. Mcs. (UFERSA)

Presidente

Cecílio Martins de Sousa Neto

Cecílio Martins de Sousa Neto, Prof. Dr. (UFERSA)

Membro Examinador

Ádller de Oliveira Guimarães

Adller De Oliveira Guimarães, Prof. Dr. (UFERSA)

Membro Examinador

“Alguns homens veem as coisas como são, e dizem ‘Por quê? ’ Eu sonho com as coisas que nunca foram e digo ‘Por que não?” (Geroge Bernard Shaw).

AGRADECIMENTOS

Agradeço primeiramente a Deus por me guiar nesta jornada e não me deixar fraquejar, por me dar sabedoria nos momentos difíceis, e por colocar as pessoas certas nos momentos certos, me ajudando a traçar uma jornada longa rumo a conquista dos meus sonhos.

A minha mãe Maria Valdênia por me incentivar sempre, e doar todo seu esforço e tempo para o melhor dos seus filhos sem pedir nada em troca, por mostrar que apesar das dificuldades haverá sempre uma esperança quando se crer em Deus, que a força de vontade pode mover montanhas. Agradeço a minha mãe por ser meu porto seguro.

A minha irmã Brenda Nayara, por sempre acreditar no meu potencial, elogiando sempre a cada feito meu, sempre me incentivando a ser melhor a cada dia.

Ao meu padrasto Marcos que sempre me incentivou a buscar o melhor que a vida pode oferecer, e a toda a minha família.

Ao meu orientador Pedro, que diante das dificuldades soube me guiar e tirar as minhas dúvidas, pela sua dedicação e disponibilidade do seu tempo, e principalmente pela paciência.

Ao meu co-orientador Lenardo, pela disponibilidade do seu tempo, e pelo auxílio em manter a qualidade deste trabalho.

A todos os meus amigos que dedicaram tempo para me ajudar quando mais precisei neste trabalho, agradeço a Allan Miqueias, um dos meus amigos que me deu apoio diante das adversidades.

RESUMO

A tecnologia está presente na vida cotidiana de milhões de pessoas em todo o mundo, com isso, o crescente uso da *Internet* como meio de comunicação faz com que uma grande quantidade de dados viaje diariamente para satisfazer os desejos das pessoas. Dessa forma, é necessário utilizar um local adequado que tenha capacidade para armazenar todos os dados que trafegam na rede. E com isso vieram os grandes *data centers*, que são salas que abrigam vários servidores cuja função é armazenar dados, processos e informações. Isso levanta questões sobre a segurança das informações que os servidores armazenam, pois é extremamente importante manter a confidencialidade dos dados e o ambiente físico. Com base nisto este projeto visa desenvolver um sistema programável de baixo custo para monitorar *data centers*. Para isso, foi utilizada a plataforma Arduino®, além de alguns sensores que possibilitam a medição de variáveis como: temperatura do ar, temperatura interna das máquinas, umidade relativa do ar, corrente elétrica, tensão elétrica , gases tóxicos e sensor de presença, o sistema também permite o controle da temperatura do ar condicionado através de pulsos infravermelhos. O presente trabalho também demonstra a construção de uma interface gráfica usando a linguagem Java e o balanço para o desenvolvimento da interface que se comunica com o Arduino® por meio de comunicação serial, podendo gerar gráficos em tempo real, enviar alertas por e-mail e visualizar as leituras através da interface, bem como uma tabela com todas as leituras armazenadas em um banco de dados local. Para garantir que não haja perda de informações capturadas pelos sensores, o sistema, além de mostrar os dados na interface gráfica, é possível visualizar as mesmas leituras através de um display LCD.

Palavras-chave: Arduino, Sistema de Medição, Interface Gráfica, Banco de Dados, Monitoramento.

ABSTRACT

Technology is present in the daily lives of millions of people around the world, with this, the increasing use of the Internet as a means of communication means that a large amount of data travels daily to satisfy people's desires. Thus, it is necessary to use a suitable location that has the capacity to store all the data that travel on the network. And with that came the big data centers, which are rooms that house several servers whose function is to store data, processes and information. This raises questions about the security of the information that servers store, as it is extremely important to maintain data confidentiality and the physical environment. Based on this, this project aims to develop a low cost programmable system to monitor data centers. For this, the Arduino® platform was used, in addition to some sensors that allow the measurement of variables such as: air temperature, internal temperature of the machines, relative humidity of the air, electric current, electrical voltage, toxic gases and presence sensor, the system also allows the control of the air conditioning temperature through infrared pulses. The present work also demonstrates the construction of a graphical interface using the Java language and the balance for the development of the interface that communicates with Arduino® through serial communication, being able to generate graphics in real time, send alerts by email and view readings through the interface, as well as a table with all readings stored in a local database. To ensure that there is no loss of information captured by the sensors, the system, in addition to showing the data on the graphical interface, is possible to view the same readings through an LCD display.

Keywords: Arduino, Measurement System, Graphical Interface, Database, Monitoring

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CH4	Metano
CO	Monóxido de carbono
CO2	Dióxido de carbono
H2	Hidrogênio
NH4	Amônia
V	Volts
IEEE	*Institute of Electrical and Electronic Engineers*
mA	Miliampére
A	Ampére
R$	Real
µA	Microampère
ppm	Partes por milhão
UR	Umidade relativa

Sumário

1 INTRODUÇÃO

Muitas das nossas atividades diárias como pesquisas na internet, uso de aparelhos portáteis, reservas em hotéis, compras *online* entre outros tipos de pesquisas, exigem a utilização de um banco de dados remoto e uma infraestrutura computacional composta por grandes servidores localizados em diversas áreas do mundo (MADHUSUDANE e SCHMIDT, 2010).

Os servidores estão concentrados em *data centers*, que nada mais é que um centro de processamento de dados, que tem como finalidade abrigar diversas quantidades de servidores e bancos de dados e processar grande quantidade de informações. O que se torna indispensável para o funcionamento adequado da economia do mundo, assim como também a vida da maioria das cidades depende do bom funcionamento e da disponibilidade de um ou vários data centers (COMSTOR, 2013). Desta maneira torna-se necessário um sistema de monitoramento para os *data centers*, pois as ameaças aos *data centers* podem resultar na perda ou degradação de alguns ou de todos os objetivos de segurança aceitos, são eles: integridade, disponibilidade e confidencialidade. Por este motivo é necessário monitorar ativamente as condições do *rack* e da sala em que o mesmo se encontra (MACEDO, 2017).

No mercado atual existem diversos dispositivos que realizam o monitoramento e gerenciamento do ambiente onde se encontra os *racks* principais. Como por exemplo, o sistema de monitoramento de *data centers* da empresa Paessler que oferece um sistema já consolidado no mercado atual atuando no monitoramento de sala de servidores, com a diferença de que o mesmo, oferece uma grande quantidade de sensores (PAESSLER, 2020). Porém o custo desses dispositivos é bastante elevado, inviabilizando a compra principalmente para as empresas de pequeno porte (YAMANOUE, 2017).

Com isso, neste trabalho é proposto o desenvolvimento de um sistema computacional para monitorar *data center* que seja de baixo custo, cujo objetivo principal é monitorar as seguintes variáveis:

- Temperatura do ar.
- Umidade relativa do ar.
- Temperatura interna do *rack*.
- Tensão elétrica.
- Corrente elétrica.

- Presença de pessoal no ambiente.
- Gases tóxicos.

O sistema projetado deve exibir as grandezas medidas por meio de uma interface gráfica, de forma que os dados possam ser visualizados por meios de gráficos individuais para cada sensor, bem como o armazenamento dos dados em um banco de dados local para que posteriormente possa ser consultado pelo administrador do sistema.

1.1. Justificativa

A importância do monitoramento do *data center* é fundamental para evitar paralisações das operações ou qualquer acesso físico de pessoas não autorizadas, o que pode causar danos ao sistema de escala global, pois o mundo é completamente dependente da tecnologia, e com tantas informações circulando na rede 24 horas, os servidores estão no centro desse armazenamento de dados. (CORREIA, 2018).

Os servidores são equipamentos que passam por picos de temperatura no ambiente instalado, assim como também sofre com a umidade excessiva. Com isso, ter um sistema que informe o aumento da temperatura, umidade relativa do ar bem como o fluxo de energia, facilita o trabalho da equipe de TI (Tecnologia da Informação) e impede a flutuação do sistema interno da empresa. Sendo assim, ter uma sala de servidor exige um acompanhamento especializado, pois quanto maior o processamento de dados, maior será a temperatura nos servidores. Deste modo, em um *data center* é essencial monitorar a temperatura, umidade, fluxo de ar, ativos, servidores, controle de acesso, segurança e outros mecanismos ABNT[1] (FURUKAWA, 2005),

A utilização deste sistema, além de garantir os requisitos mínimos de segurança necessários para o funcionamento dos *data centers*, pode levar um aumento significativo na sua vida útil, o que acarreta menores custos para as empresas.

[1] http://www.itbusinessday.com.br/2010/conteudo/sergio.pdf

1.2 Objetivos

Neste capitulo serão apresentados os objetivos gerais e específicos relacionado a proposta deste trabalho de conclusão de curso.

1.2.3 Objetivo Geral

Desenvolver um sistema de monitoramento de baixo custo a partir de uma API (*Application Programing Interface*) utilizando a plataforma Arduino® e a linguagem de programação Java para as devidas manipulações dos dados coletados através dos sensores externos, e com isso poder compartilhar essas informações através de uma interface gráfica.

1.2.4 Objetivos Específicos

- Desenvolver uma API que auxilie o usuário final no monitoramento das salas de servidores
- Realizar estudos dos sensores para melhor utilização dos mesmos no projeto
- Desenvolver um sistema de controle da temperatura do ar condicionado

1.3 METODOLOGIA DE PESQUISA

Como descrito anteriormente, o objetivo é criar um sistema de monitoramento para sala de servidores utilizando a plataforma Arduino® e a linguagem de programação Java para o desenvolvimento da interface gráfica. Logo, elaboram-se as seguintes questões de pesquisa a seguir (QP):

- Quais as principais necessidades a serem atendidas com o sistema de monitoramento?
- Quais as funcionalidades/opções, devem ser implementadas no sistema para que seja útil as demandas dos usuários?

As questões abordadas serão utilizadas para influenciar, analisar e obter resultados deste trabalho, e com isso auxiliar no seu desenvolvimento.

1.4 ORGANZAÇÃO DO DOCUMENTO

Este documento está estruturado em 4 capítulos além da introdução. No capítulo 2 é abordado um estudo mais aprofundado sobre os sensores que compõe este trabalho, bem como o software que irá se comunicar com o usuário final e outros assuntos que serão fundamentais para a explanação e o desenvolvimento deste trabalho. No capítulo 3 será abordada a metodologia de teste experimental dos sensores e da aplicação desenvolvida cujo objetivo é apresentar resultados que que foram obtidos através de uma série de testes com equipamentos já consolidados no mercado. No capítulo 4 é apresentada a conclusão sobre o trabalho, trabalhos futuros, de acordo com a pesquisa, seu resultado final e limitações.

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentados os sensores utilizados no sistema.

2.1 Sensores de gases tóxicos encontrados no mercado

No mercado atual, existem diversos modelos de sensores capazes de medir a concentração de gases tóxicos e inflamáveis, assim como também existem outros sensores que realizam apenas a detecção desses gases.

O Quadro 1 apresenta um levantamento de tipos, fabricante e preço de sensores levando em consideração que os preços estão em dólar se baseando nas informações apresentadas pela UOL Economia (2020).

Quadro 1: Sensores e detectores de gases no mercado atual

Tipo do sensor	**Marca**	**Preço ($)**
Detector de Monóxido de Carbono Co de precisão	BENETECH	186,96
Detector de Hidrogênio	Instrutherm	497,68
Detector de GLP/ Gás natural	Gaseg	64,70
Detector de Metano	SAIFULI	29,29
Detector de GLP portátil	Samtek	14,18
Detector de fumaça	Segurimax	8,75
Detector de Gases Inflamáveis MQ-2	*Hanwei Eletronics*	1,19

Fonte: Mercado livre (www.mercadolivre.com, acesso em 3 de jan. 2020, 14:30) / Ali Express (www.aliexpress.com, acesso em 3 de jan. 2020, 14:31)

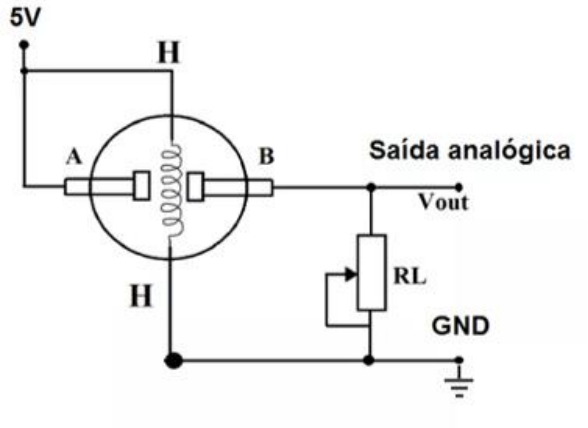
Com isso, em busca de maior custo benéfico, e qualidade, também existe os sensores da família MQ-X, da empresa *Hanwei Eletronics*, na qual com apenas um único sensor, é possível monitorar vários tipos de concentração de gases no ambiente, o que se torna essencial para o presente trabalho.

2.2 Sensor de gás da família MQ-X

Os sensores da família MQ-X são capazes de detectar a concentração de diversos gases tóxicos em um determinado ambiente tais como: fumaça, dióxido de carbono, benzeno, óxido nítrico, álcool, hidrogênio, (gás liquefeito de petróleo) GLP, metano e monóxido de carbono (ELETRONICS, 2020).

Estes sensores são eletroquímicos e variam sua resistência quando expostos a certos tipos de gases, apresentando em seu circuito interno um aquecedor que é o responsável por aumentar a temperatura do sensor. Desta forma o sensor pode reagir a diferentes tipos de gases pois o circuito básico para o funcionamento do sensor MQ-X é baseado em divisor de tensão conforme é ilustrado na Figura 1(CANDIDO, 2017).

Figura 1: Circuito equivalente do sensor da família MQ-X



Fonte: (CANDIDO, 2017)

De A para B, temos o sensor de gás, em um ambiente que possa conter gás poluidor, a resistência do sensor de gás diminui conforme a concentração do gás poluente aumenta. Ou seja, quanto mais gás poluente, menor será a resistência entre A e B, e com isso maior será a tensão em cima do resistor de carga RL (CANDIDO, 2017).

É importante tomar nota que o aquecedor permite que o sensor esteja em uma temperatura ideal para que o mesmo possa realizar as leituras mais precisa possível, desta forma, é necessário um tempo de aquecimento denominado *Burn-in time* (Tempo de queima) ou *preheat* (pré-aquecimento). Esse tempo varia de modelo para modelo, (CANDIDO, 2017).

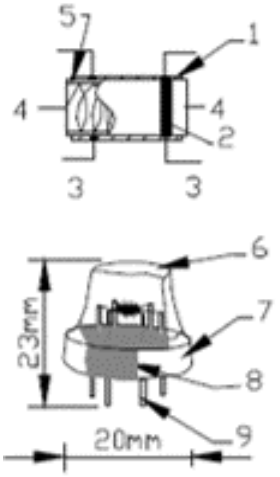
O Quadro 2, demonstra como está dividido as partes que compõem os sensores da família MQ-X bem como o material a qual os mesmos são construídos. A distribuição dos componentes pode ser vista na Figura 2 (ELETRONICS, 2020).

Quadro 2: Componentes dos sensores da família MQ-X

	Partes	**Material**
1	Camada sensitiva	Óxido de Estanho (SnO2)
2	Eletrodo	Cobre (Cu)
3	Linhas do eletrodo	Platina (Pt)
4	Bobina do aquecedor	Liga de Níquel-Cromo (Ni-Cr)
5	Tubo de cerâmica	Óxido de alumínio (Al2O3)
6	Trançado anti explosão	Aço inoxidável
7	Anel de fixação	Cobre e Níquel
8	Base de resina	Baquelite
9	Pinos	Cobre e Níquel

Fonte: *Datasheet* do sensor MQ-2

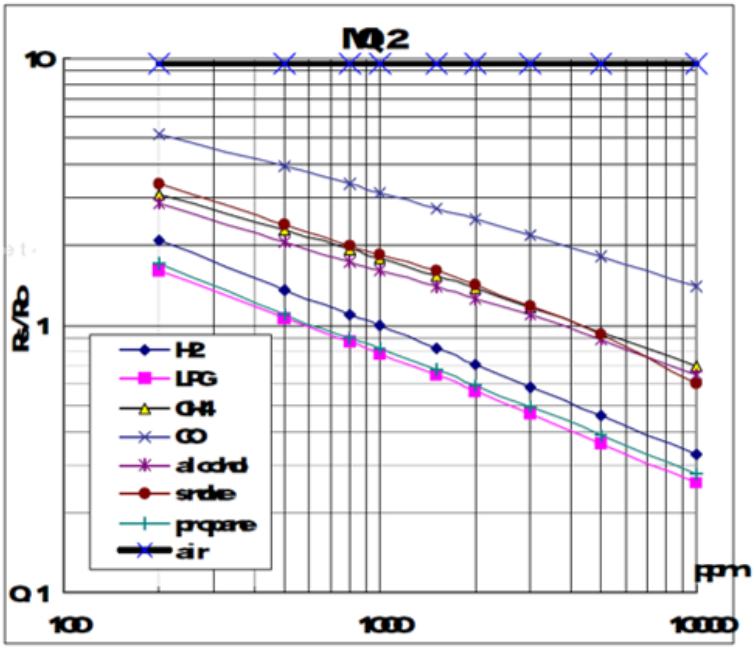
Figura 2: Distribuição dos componentes dos sensores da família MQ-X



Fonte: *Datasheet* do sensor MQ-2

O sensor MQ-2 foi escolhido para este projeto dado que este possui uma boa sensibilidade para vários tipos de gases tóxicos, como por exemplo: álcool, fumaça, metano, GLP, monóxido de carbono, e hidrogênio. Na Figura 3, pode ser ilustrado ver que o sensor reage de maneira diferente a vários tipos de gases. É demonstrado também que o sensor apresenta alta resistência ao ar limpo, e por este motivo, devido a sua alta sensibilidade, pode-se detectar uma concentração de gases de forma eficaz e com alta velocidade de resposta ao microcontrolador.

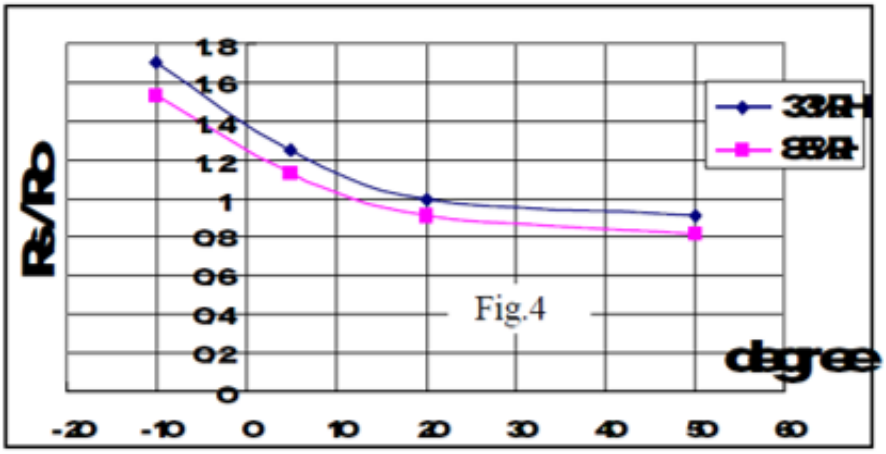
Figura 3: Resposta do sensor MQ-2 à diferentes gases



Fonte: (ELETRONICS, 2020)

Na Figura 4, o fabricante demonstra que o desempenho dos sensores da família MQ-x são influenciados diretamente pela temperatura e pela umidade relativa do ar, e por este motivo, necessita-se de correções em suas leituras. Porém como mencionado na seção 2.1 o ajuste não será necessário, visto que segundo a norma que rege a sala de servidores ANSI/TIA-942, a temperatura do ambiente tem que está em torno de 20 a 25ºC e a humidade relativa do ar em torno de 40 a 50%, o que segundo o datasheet do fabricante são medidas aceitáveis para uma boa leitura (ELETRONICS, 2020).

Figura 4: Influência da humidade e temperatura no sensor MQ-2

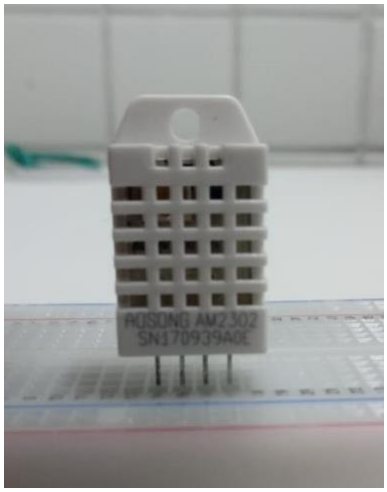


Fonte: (ELETRONICS, 2020)

Diante do exposto conclui-se que a aplicabilidade do sensor MQ-2 está em situações em que as leituras estejam entre 200 a 10000 ppm e com temperatura e umidade ideal, o que neste projeto o mesmo se torna eficaz, pelo seu baixo custo de mercado, e pelo seu desempenho dentro em um ambiente controlado.

2.3 Sensor DHT22

Figura 5: Sensor DHT22



Fonte: Autoria própria

Como a sala dos servidores necessita do monitoramento da temperatura e da umidade, é necessário que haja o devido controle dessas variáveis. Pensando nisso foi escolhido para este projeto o sensor DHT22 para realizar as medições (ELECTRONICS, 2020).

O sensor DHT22 é um sensor que realiza as leituras da temperatura e umidade com uma precisão de 0,5ºC e 2% UR respectivamente, possuindo um tempo de resposta de 2 segundos. Além de ser também um sensor de baixo custo, o seu preço pode ser visto no Quadro 9 (MURTA, 2020). Os dados do sensor podem ser encontrados no Quadro 3 e seus aspectos físicos na Figura 5.

Quadro 3: Dados sobre o sensor DHT22

Tensão de entrada	3,3-6 V DC
Tipo de saída	Sinal digital
Alcance de operação para umidade	0 – 100%UR
Alcance de operação para temperatura	-40 – 80ºC
Precisão para umidade	+/- 2% UR
Precisão para temperatura	+/- 0,5 ºC
Resolução para umidade	0,1 %UR
Resolução para temperatura	0,1 ºC
Tempo de medição	2 s

Fonte: Electronics (2020)

O sensor DHT22, por ser um sensor duplo, tem uma grande vantagem no que diz respeito à economia de pinos do processador do Arduino®, pois o mesmo realiza leituras da umidade relativa do ar e da temperatura ambiente com apenas um pino digital do processador, assim como os sensores da família MQ-x mencionados na seção 2.1. Por este sensor utilizar uma porta digital para realizar as suas leituras, torna-se mais precisa a sua leitura por ser menos susceptível a flutuações nas leituras, bem como a sua facilidade de uso.

2.4 Arduino® Nano

Figura 6: Arduino® Nano

Fonte: Autoria própria

A plataforma Arduino® é uma plataforma de desenvolvimento de baixo custo e de fácil acesso, sendo composto por: A placa, e a IDE, que é o *software* onde escrevemos as instruções que desejamos.

A maior vantagem dessa plataforma de desenvolvimento sobre as demais é a facilidade de uso. Dentro desta plataforma existem diversos tipos de Arduino®, como, por exemplo: Nano, Uno e Mega. A versão Nano foi escolhida para este projeto por ter um tamanho reduzido, facilitando o acoplamento em uma *protoboard*. As suas características podem ser vistas no Quadro 4 e a sua estrutura física na Figura 6.

Quadro 4: Características do Arduino® Nano

Microcontrolador	Atmel ATmega168 ou ATmega328
Voltagem de operação (nível logico)	5 V
Voltagem de entrada (recomendada)	7-12 V
Voltagem de entrada (limites)	6-20 V
Pinos digitais I/O	14 (dos quais 6 podem ser saídas PWM)
Pinos de entrada analógica	8
Corrente de entrada analógica	40 mA

Memória Flash	16 KB (ATmega168) ou 32 KB (ATmega368)
SRAM	1 KB (ATmega168) ou 2 KB (ATmega368)
EEPROM	512 bytes (ATmega168) ou 1 KB (ATmega368)
Velocidade de Clock	16 MHz
Dimensões	0.73”x 1.70”

Fonte: Comphaus (2020)

2.5 Sensor de corrente não invasivo 30A SCT-013

Figura 7: Sensor 30A SCT-013

Fonte: Autoria Própria

Monitorar a corrente elétrica é de suma importância para manter o funcionamento aduado dos servidores, pois através do consumo das máquinas é possível detectar possíveis problemas eletrônicos ocasionado por componentes danificados. Visando confiabilidade e baixo custo, foi escolhido para este projeto o sensor e corrente não invasivo 30ª SCT-013.

As características do sensor 30A SCT-013 podem ser vistas no Quadro 5 e sua estrutura física na Figura 7.

Quadro 5: Característica do sensor 30A SCT-013

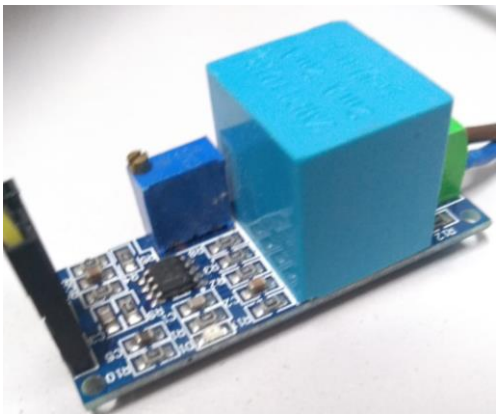
Modelo:	SCT-013-020

Corrente de entrada	0-20A
Sinal de saída	Tensão/1V
Material do Core	Ferrite
Dielétrico	6000V AC/1min
Taxa anti-chama	UL94-V0
Plug de saída	3,5mm
Dimensão de abertura	13 x 13mm
Temperatura de trabalho	-25 a + 70ºC
Comprimento do cabo	150cm

Fonte: Filipiflop (2020)

2.6 Sensor de tensão AC ZMPTB

Figura 8: Sensor de tensão AC ZMPT010B



Fonte: Autoria própria

Assim como monitorar a corrente elétrica, é importante monitorar a tensão da rede elétrica, pensando nisso, o sensor de tensão AC ZMPTB foi o escolhido, pois o mesmo é de baixo custo e cumpre uma função relevante para o monitoramento dos picos de alta tensão que por sua vez que podem ocasionar danos as máquinas.

O sensor de tensão AC ZMPTB é um módulo de alta precisão que tem como finalidade detectar a existência de tensão alternada em um circuito podendo ser utilizado como um

voltímetro ligando-o em paralelo com a rede elétrica. As suas características podem ser vistas no Quadro 6 e suas características físicas na Figura 8.

Quadro 6: Características do sensor de tensão AC ZMPTB

Transformador	ZMPT010B
Tipo de sensor	Detector de tensão / voltímetro
Tensão de alimentação	5 a 30VDC
Tensão de entrada	0 a 250VCA
Corrente de entrada nominal	2mA
Corrente de saída nominal	2mA
Proporção	1000:1000
Faixa linear	0-1000V
Linearidade	0,2%
Isolamento tensão	4000V
Precisão de leitura	+-1%
Temperatura de operação	-40º a 70º Celsius
Dimensões	22mm (L) X 20mm (A) X 51mm (C)
Peso	20g

Fonte: Masterwalker (2020)

2.6 Sensor de presença PIR DYP-ME003

Figura 9: Sensor de presença PIR DYP-ME003

Fonte: Autoria própria

A sala de servidores é um ambiente que deve ser monitorado o acesso de pessoal não autorizado pelo fato de que na mesma contém informações importantes que são armazenadas constantemente, e com isso ter o controle do acesso a esses ambientes é de extrema importância, pesando nisso, neste projeto, foi escolhido o sensor de presença PIR DYP-ME003.

Este sensor é encontrado facilmente no mercado atual de baixo custo. Ele consegue detectar o movimento de objetos que estejam em uma área de até 7 metros. Uma outra vantagem, é a possibilidade de calibrar o tempo de espera para que o pino digital se mantenha em ativo, assim como também permite a calibração da sensibilidade do sensor. As suas características podem ser vistas no Quadro 7 e sua estrutura física na Figura 9.

Quadro 7: Características do sensor PIR DYP-ME003

Modelo	DYP-ME003
Sensor infravermelho com controle na placa	
Sensibilidade e tempo ajustável	
Tensão de operação	4,5-20V
Tensão de dados	3,3V (Alto) – 0V (Baixo)
Distância detectável	3-7m (Ajustável)
Tempo de delay	5-200seg (Default: 5seg)
Tempo de bloqueio	2,5seg (Default)

Trigger	(L) -Não Repetível (H) -Repetível (Default: H)
Temperatura de trabalho	-20 ~+80ºC
Dimensões	3,2 x 2,4 x 1,8cm
Peso	7g

Fonte: Filipiflop (2020)

2.8 Sensor de temperatura Max 6675 (Termopar)

Figura 10: Sensor de temperatura (Termopar)

Fonte: Autoria própria

Assim como monitorar a temperatura do ambiente em uma sala de servidores é extremamente importante também é necessário monitorar a temperatura interna dos processadores pois alto aquecimento nos mesmos poderá ocasionar a paralisação de todo o *hardware*.

Pensando nisto, o escolhido para este projeto foi o termopar Max 6675. Sendo este um sensor de baixo custo, e de fácil acesso, além de resistir a temperaturas segundo o fabricante de 0 a 800ºC. Além disso o mesmo é blindado contra umidade, e temperatura, o que o torna útil para monitorar a temperatura interna das máquinas. As suas características podem ser vistas no Quadro 8 e sua estrutura física na Figura 10.

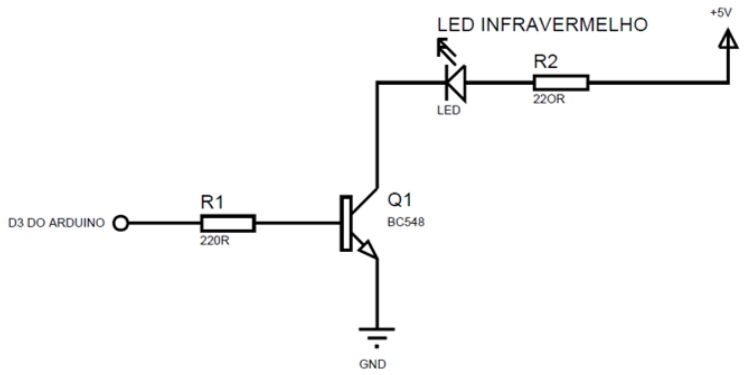
Quadro 8: Características do sensor de temperatura MAX6675

CI	MAX6675
Tensão de trabalho	5V
Corrente de trabalho	50mA
Precisão	~1,5%
Termopar	Blindado
Possibilidade de comunicação tipo	SPI
Temperatura do termopar	0 a 800°C
Comprimento do termopar	50cm
Cabo externo blindagem	Stainless Setel Brainding
Rosca	M8

Fonte: Eletrônica (2020)

2.9 Circuito de controle do ar condicionado

Figura 11: Características do circuito que controla o ar condicionado.



Fonte: Autoria própria

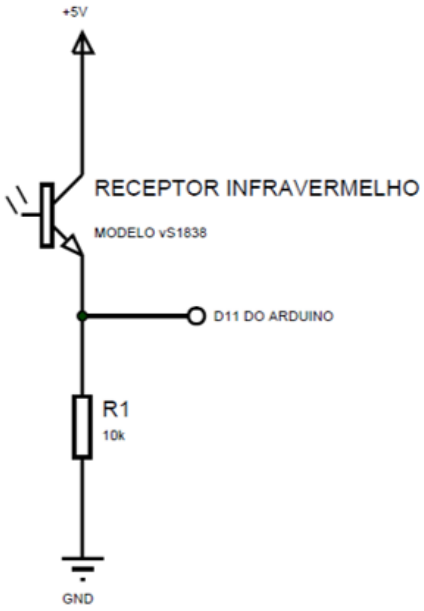
Como a sala de servidores é uma área de pouco acesso, esta passa maior parte do tempo sem a presença de pessoas, e com isso levanta-se a questão do controle do ar condicionado para manter o ambiente sempre em valores ideais para um bom funcionamento dos servidores. Diante disso, se por algum motivo a temperatura atinja níveis perigosos é necessário um controle do ar condicionado para diminuir a temperatura do ambiente e manter a sala na temperatura ideal.

Pensando nisso, o sistema terá que ser capaz de controlar o ar condicionado, de forma automática, necessitando de um circuito de controle utilizando infravermelhos foi escolhido para esta tarefa. O circuito que realiza tal tarefa é visto na Figura 11.

O Arduino® enviara um sinal para acionar o circuito mostrado na Figura 11, que em seguida irá ajustar a temperatura do ar condicionado, analisando a Figura 11 nota-se que que o sinal não foi injetado diretamente no LED infravermelho, pois foi necessária uma etapa de potência para amplificar a corrente de acionamento do LED infravermelho, para que o mesmo possa alcançar maiores distâncias. Com isso o transistor BC548 realiza a tarefa de amplificar a corrente do led.

O sinal enviado pelo Arduino® é um sinal de onda quadrada que faz com que o LED infravermelho ligue e desligue diversas vezes com uma frequência determinada, fazendo com que o aparelho de ar condicionado reconheça os pulsos gerados e execute uma determinada ação. Com isso, para que o Arduino® possa enviar os pulsos referentes ao ar condicionado, é necessário um circuito receptor de infravermelho, para que possamos clonar o controle do ar condicionado que irá gerar o sinal do ar condicionado, e com isso o Arduino® irá reproduzir o mesmo sinal. As características da montagem desse circuito receptor de sinal infravermelho podem ser vistas na Figura 12.

Figura 12: Características do circuito receptor de sinal infravermelho



Autor: Autoria própria

2.10 Componentes

Visto que o objetivo deste projeto é confeccionar um sistema de monitoramento para sala de servidores que seja de baixo custo, o Quadro 9 demonstra cada componente utilizado e seu preço respectivamente, obtendo como custo total R$ 167,92. No mercado atual, existem soluções para o monitoramento de sala de servidores como, por exemplo, da empresa Paessler[2] cujo segundo a empresa, disponibiliza em seu site a lista de preços para se adquirir o sistema de monitoramento cujo preço mais baixo para monitorar apenas um servidor é em torno de R$ 1200,00 (PAESSLER, 2020).

Conforme a solicitação do cliente para o monitoramento de dois ou mais servidores o preço pode chegar, aproximadamente, a R$ 60.000,00.

Quadro 9: Componentes do sistema de monitoramento

Quantidade	Componentes	Preço (R$)
1	Arduino® Nano	3,75
1	Sensor de temperatura e humidade DHT22	44,91
1	Sensor de Gás MQ-2	5,17
1	Sensor de corrente elétrica 30A SCT-013	15,55
1	Sensor de tensão elétrica AC ZMPT010B	26,00
1	Sensor de temperatura termopar AC ZMPT010B	20,46
1	Sensor de presença PIR DYP-ME003	11,90
1	Display LCD 20x4	36,90
1	Módulo 12C para o display LCD	3,28
	Total (R$)	167,92

Fonte: Autoria própria

2.11 Montagem

No Quadro 10 está listado as ligações físicas entre os sensores descritos nesta seção e o Arduino®.

[2] https://www.br.paessler.com/server_room_monitoring

Quadro 10: Ligações físicas entre o Arduino e os sensores

Origem	Arduino®
Pino A0 do MQ-2	Pino analógico 3
Pino VCC do MQ-2	Alimentação por fonte de 5V externa
Pino GND do MQ-2	GND (Terra) do Arduino® e da fonte externa
Pino 1 do DHT22	Alimentação por fonte de 5 V externa
Pino 2 do DHT22	Pino digital 10
Pino 3 do DHT22	NULL
Pino 4 do DHT22	GND (Terra) do Arduino® e da fonte externa
Pino GND do termopar	GND (Terra) do Arduino® e da fonte externa
Pino VCC do termopar	Alimentação por fonte de 5 V externa
Pino SCK do termopar	Pino digital 7
Pino CS do termopar	Pino digital 6
Pino SO do termopar	Pino digital 5
Pino 2 do sensor de presença (OUT)	Pino digital 8
Pino GND do sensor de presença	GND (Terra) do Arduino® e da fonte externa
Pino de VCC do sensor de presença	Alimentação por fonte de 5 V externa
Pino GND do sensor de tensão	GND (Terra) do Arduino® e da fonte externa
Pino VCC do sensor de tensão	Alimentação por fonte de 5 V externa
Pino OUT do sensor de tensão	Pino analógico 2
Pino INP do sensor de corrente	Pino analógico 0
Pino GND do módulo I2C (LCD)	GND (Terra) do Arduino® e da fonte externa
Pino VCC do módulo I2C (LCD)	Alimentação por fonte de 5 V externa
Pino SDA do módulo I2C (LCD)	Pino analógico 4
Pino SCL do módulo I2C (LCD)	Pino analógico 5
Pino OUT LED infravermelho	Pino digital 3
Pino INP do receptor infravermelho	Pino digital 11

Fonte: Autoria própria

Neste caso, é necessário o uso de uma fonte de alimentação externa de 5V com a capacidade de corrente de no mínimo 1A.

2.12 Metodologia de medição

Para realizar as medições utilizou-se a comunicação serial entre o Arduino® e o computador que no qual o software é executado. Por tanto, para iniciar a comunicação serial, é necessário utilizar algumas DLLs, cujo procedimento de instalação pode ser visto no Apêndice A.

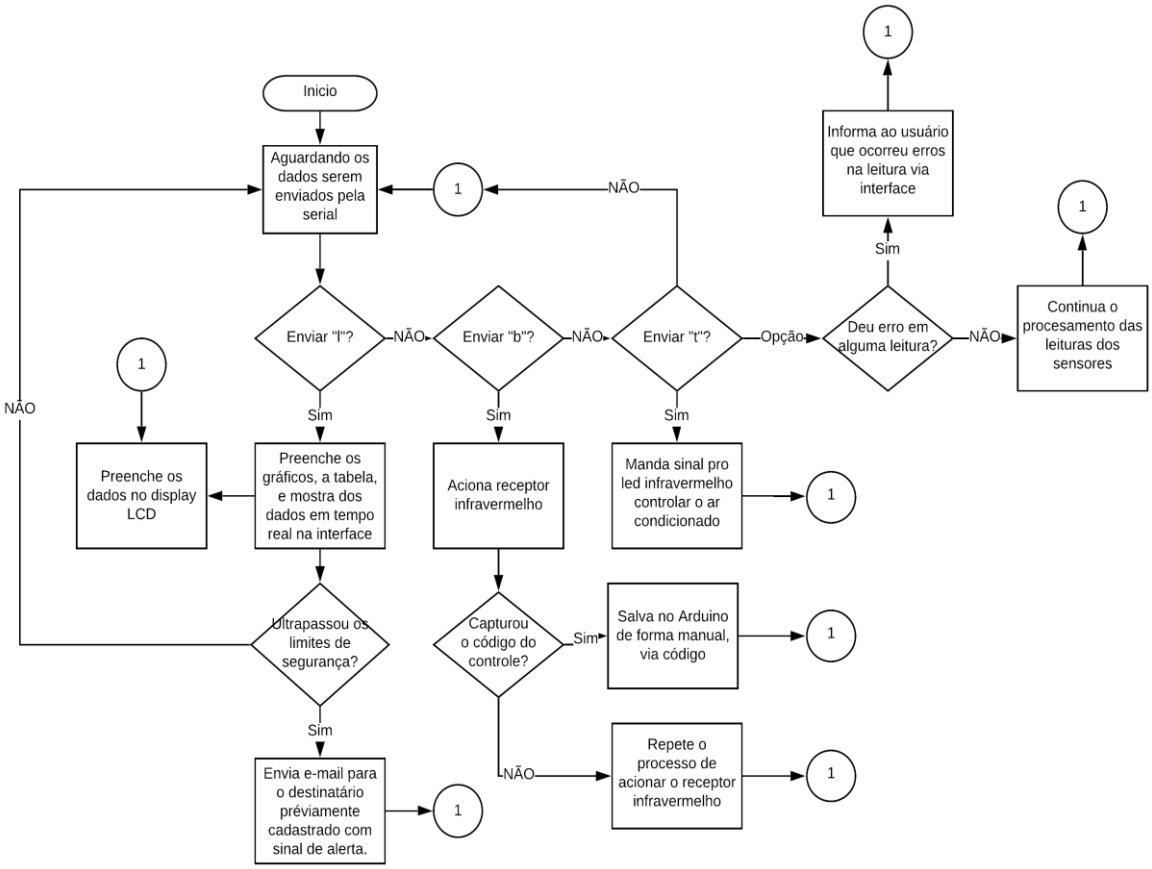
Estas DLLs, são importantes pois as mesmas permitem a comunicação serial entre o Arduino® e a interface gráfica feita com a linguagem Java. Com isso o Arduino® irá apenas coletar dados provenientes dos sensores e enviar via comunicação serial para a interface, e logo em seguida os devidos tratamentos com as informações ficam a cargo apenas da interface do projeto. Por questões segurança o sistema também conta com um display LCD 20x4 para mostrar as leituras dos sensores de forma mais rápida.

2.13 Software

Na Figura 13, pode ser visto o fluxograma do software, contendo os caminhos cujo as informações são repassadas entre a aplicação Java[3] juntamente com a sua biblioteca *swing*, e o Arduino®.

[3] https://www.java.com/pt_BR/

Figura 13: Fluxograma do Software do Sistema de Monitoramento

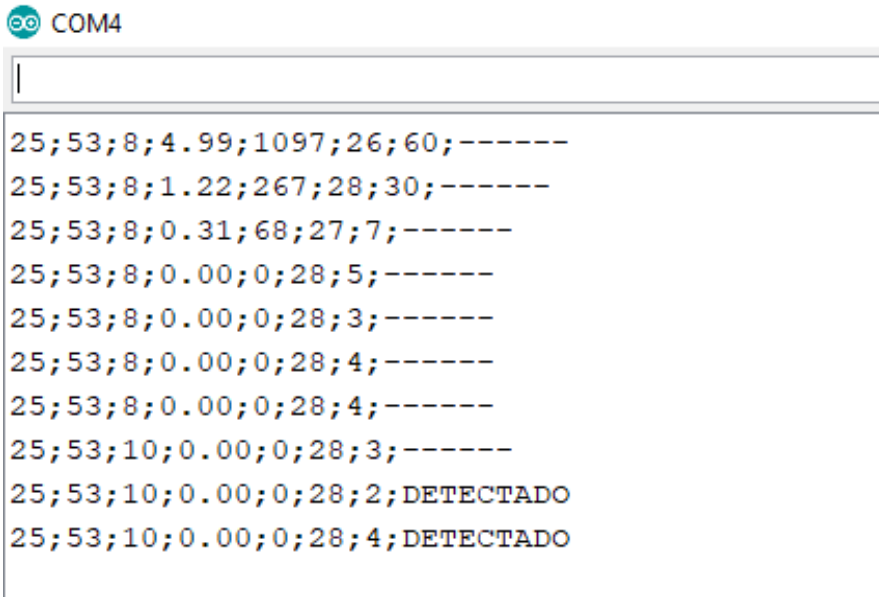


Fonte: Autoria própria

O *Software* implementado é responsável apenas por tratar as leituras que o Arduino® envia por comunicação serial. Para a construção da interface foi utilizada a linguagem de programação Java, e o swing para criação da interface. Esta etapa de leitura segue os seguintes passos:

1. Inicialmente ao conectar o Arduino® à porta USB do computador onde o sistema encontra-se em execução, o sistema irá aguardar por uma entrada até que o Arduino® envie as leituras dos sensores de forma concatenada, ou seja, toda a leitura vem em uma única *string*, exibido na Figura 14, e a aplicação se encarrega de tratar cada dado individualmente.

Figura 14: Formato da *String* enviada pelo Arduino®

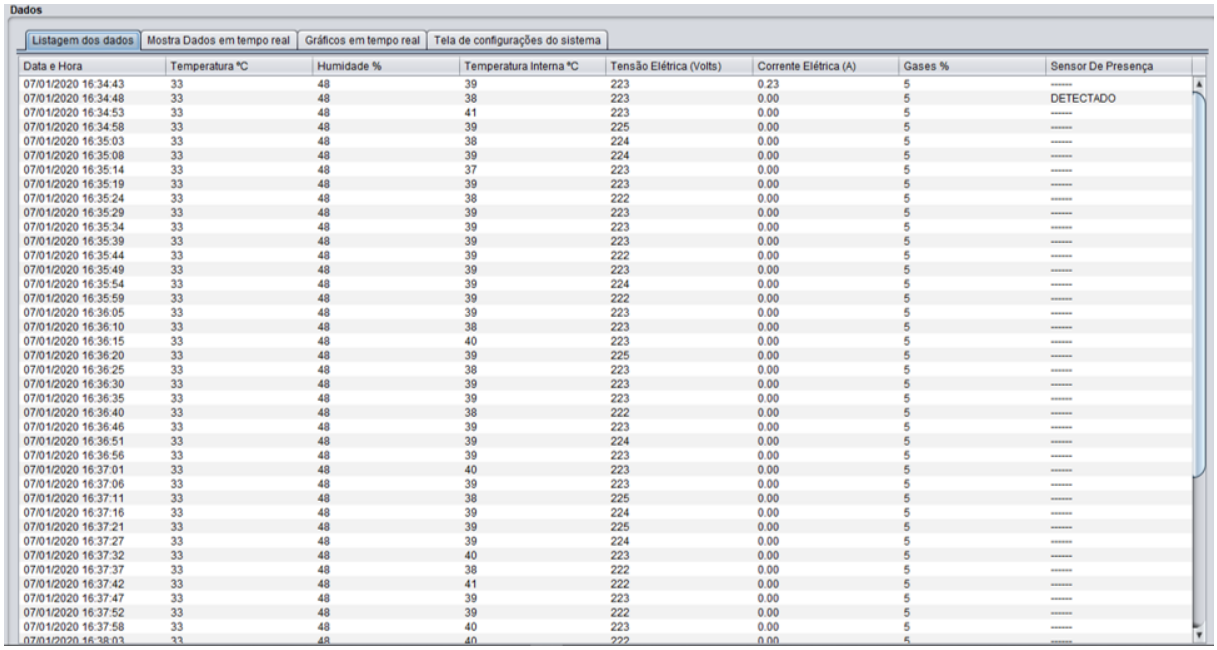


Fonte: Autoria próprio

2. Uma vez obtido a confirmação de dados sendo enviados pela serial, o *software* irá mostrar em tempo real as leituras dos sensores. Para isso, o *software* tem 3 três opções de visualização de dados:
    2.1. A primeira interface é uma tabela que armazena todas as leituras dos sensores, na qual é atualizada também em tempo real. Os dados são armazenados em um banco de dados local, utilizando a ferramenta PostgreSQL[4], e com isso as informações não são perdidas. A organização da tabela pode ser vista na Figura 15.

[4] https://www.postgresql.org/

Figura 15: Tela de Apresentação do Histórico de Dados

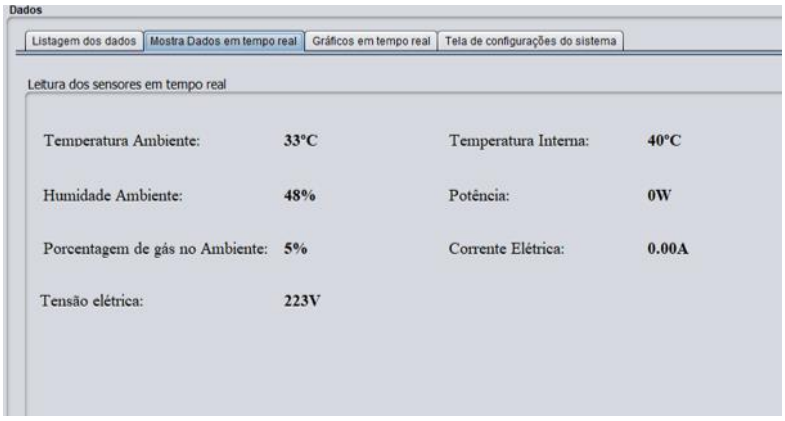


Data e Hora	Temperatura °C	Humidade %	Temperatura Interna °C	Tensão Elétrica (Volts)	Corrente Elétrica (A)	Gases %	Sensor De Presença
07/01/2020 16:34:43	33	48	39	223	0.23	5	------
07/01/2020 16:34:48	33	48	38	223	0.00	5	DETECTADO
07/01/2020 16:34:53	33	48	41	223	0.00	5	------
07/01/2020 16:34:58	33	48	39	225	0.00	5	------
07/01/2020 16:35:03	33	48	38	224	0.00	5	------
07/01/2020 16:35:08	33	48	39	224	0.00	5	------
07/01/2020 16:35:14	33	48	37	223	0.00	5	------
07/01/2020 16:35:19	33	48	39	223	0.00	5	------
07/01/2020 16:35:24	33	48	38	222	0.00	5	------
07/01/2020 16:35:29	33	48	39	223	0.00	5	------
07/01/2020 16:35:34	33	48	39	223	0.00	5	------
07/01/2020 16:35:39	33	48	39	223	0.00	5	------
07/01/2020 16:35:44	33	48	39	222	0.00	5	------
07/01/2020 16:35:49	33	48	39	223	0.00	5	------
07/01/2020 16:35:54	33	48	39	224	0.00	5	------
07/01/2020 16:35:59	33	48	39	222	0.00	5	------
07/01/2020 16:36:05	33	48	39	223	0.00	5	------
07/01/2020 16:36:10	33	48	38	223	0.00	5	------
07/01/2020 16:36:15	33	48	40	223	0.00	5	------
07/01/2020 16:36:20	33	48	39	225	0.00	5	------
07/01/2020 16:36:25	33	48	38	223	0.00	5	------
07/01/2020 16:36:30	33	48	39	223	0.00	5	------
07/01/2020 16:36:35	33	48	39	223	0.00	5	------
07/01/2020 16:36:40	33	48	38	222	0.00	5	------
07/01/2020 16:36:46	33	48	39	223	0.00	5	------
07/01/2020 16:36:51	33	48	39	224	0.00	5	------
07/01/2020 16:36:56	33	48	39	223	0.00	5	------
07/01/2020 16:37:01	33	48	40	223	0.00	5	------
07/01/2020 16:37:06	33	48	39	223	0.00	5	------
07/01/2020 16:37:11	33	48	38	225	0.00	5	------
07/01/2020 16:37:16	33	48	39	224	0.00	5	------
07/01/2020 16:37:21	33	48	39	225	0.00	5	------
07/01/2020 16:37:27	33	48	39	224	0.00	5	------
07/01/2020 16:37:32	33	48	40	223	0.00	5	------
07/01/2020 16:37:37	33	48	38	222	0.00	5	------
07/01/2020 16:37:42	33	48	41	222	0.00	5	------
07/01/2020 16:37:47	33	48	39	223	0.00	5	------
07/01/2020 16:37:52	33	48	39	222	0.00	5	------
07/01/2020 16:37:58	33	48	40	223	0.00	5	------
07/01/2020 16:38:03	33	48	40	222	0.00	5	------

Fonte: Autoria própria

2.2. Na segunda interface o *software* o usuário possa visualizar apenas as leituras em tempo real dos sensores, e verificar se há algo de errado com alguma leitura especifica. A organização desta interface pode ser vista na Figura 16.

Figura 16: Tela de Apresentação de Dados em Tempo Real (Interface II)

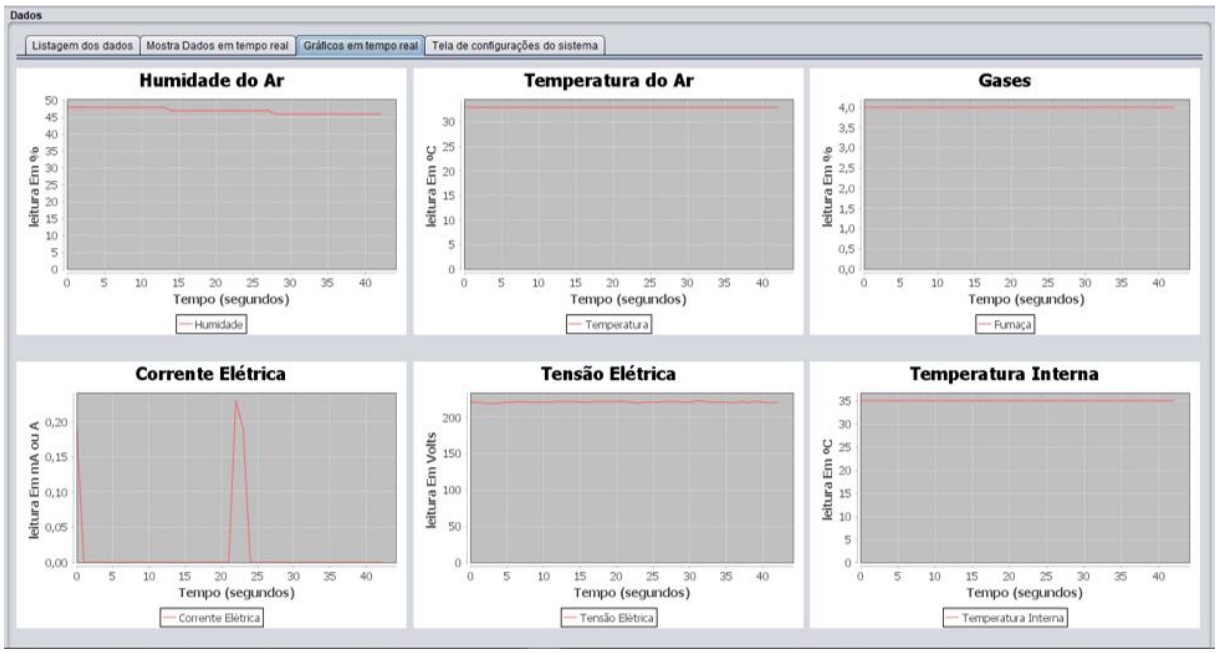


Fonte: Autoria própria

2.3. Na terceira interface, o *software*, traz uma série de gráficos, que são gerados a partir das leituras que são enviadas pela serial, e mostra para o usuário de forma gráfica como está o comportamento de cada sensor individualmente, além, de incorporar a

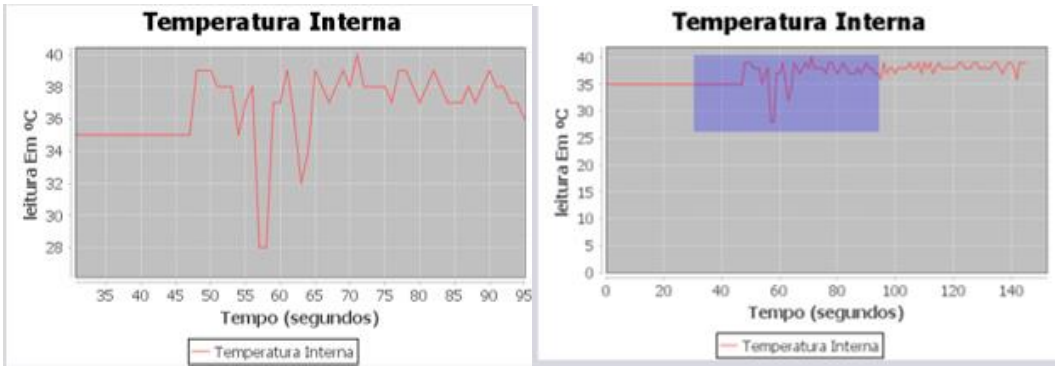
funcionalidade de zoom. As características desta interface podem ser vistas na Figura 17 e o funcionamento da ferramenta de zoom na Figura 18.

Figura 17: Tela de Apresentação dos Dados no Modo Gráfico (Interface III)



Fonte: Autoria própria

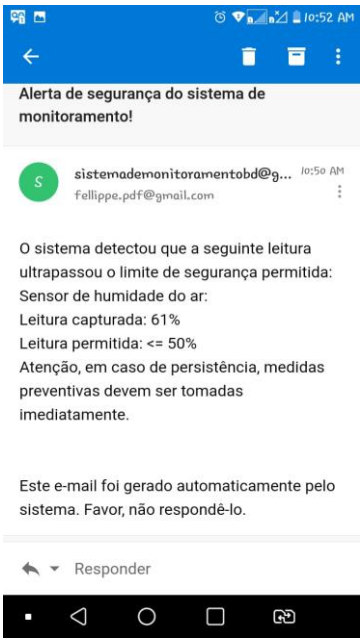
Figura 18: Ferramenta de *zoom* dos gráficos



Fonte: Autoria própria

3. Em caso de aumento considerado nas leituras dos sensores, o *software* por medida de segurança, envia um e-mail de alerta para o endereço de e-mail cadastrado no sistema informando qual sensor ultrapassou os limites considerados seguros para a segurança da sala de servidores. Esta funcionalidade pode ser vista na Figura 19.

Figura 19: Exemplo de E-mail de Segurança e Enviado para o Destinatário Previamente Cadastro no Sistema



Fonte: Autoria própria

Além da interface gráfica, que só pode ser visualizada através de um computador as leituras também podem ser visualizadas em um LCD 20x4. Conforme visto na Figura 20.

Figura 20: Apresentação da Leitura dos Sensores no display LCD

Fonte: Autoria própria

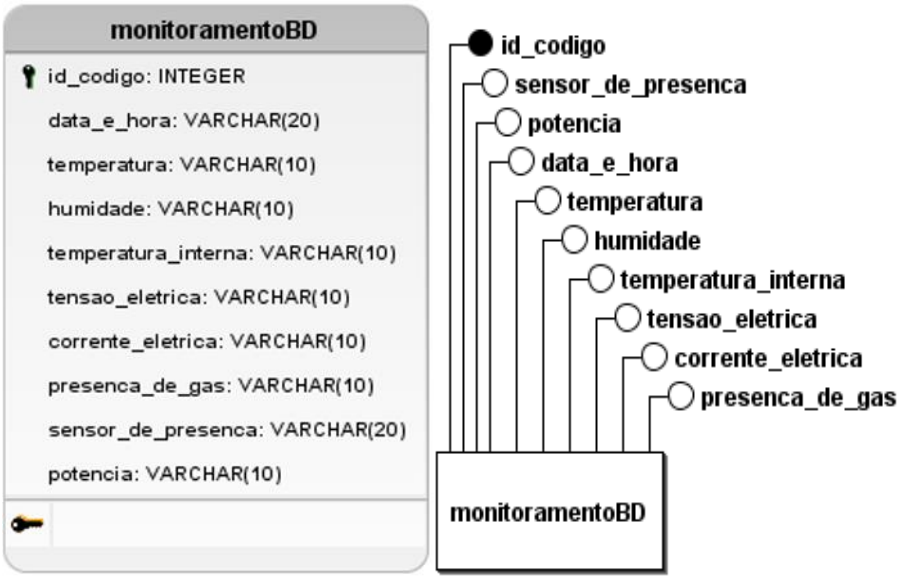
2.14 Banco de dados

Para o armazenamento dos dados capturados via sensores e enviados pelo Arduino® ao sistema foi utilizado o sistema de Gerenciamento de Banco de Dados PostgreSQL versão desktop, visto que as informações serão guardadas na própria máquina que executa a aplicação, ou seja, os dados estão a todo momento disponíveis do usuário sem a necessidade do uso da internet para o armazenamento.

Na aplicação deste projeto, o banco de dados não tem influência na geração dos gráficos, o *software* acessa o banco de dados apenas para a população de dados na *interface* de apresentação dos dados para o usuário demonstrado na Figura 17. Os gráficos deste projeto são atualizados em tempo real através da comunicação Serial entre o Arduino® e o computador que está com a aplicação.

O banco de dados do sistema possui apenas com uma única relação. O modelo lógico e conceitual do banco pode ser visto na Figura 21.

Figura 21: Modelo Lógico e conceitual do banco de dados



Fonte: Autoria própria

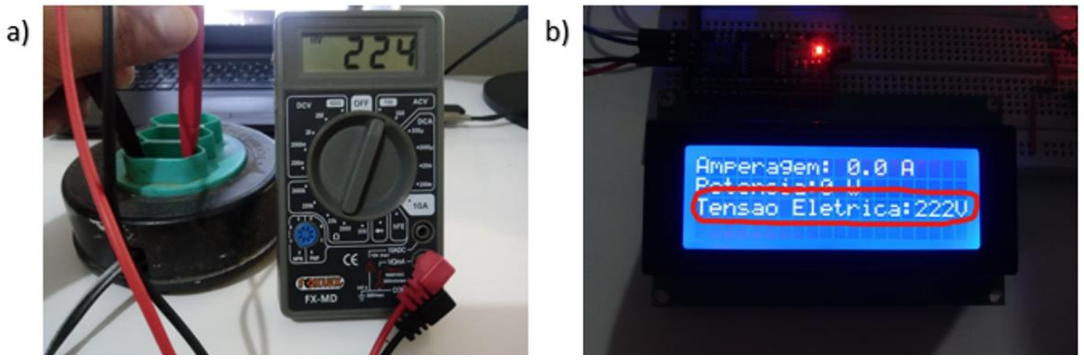
3 TESTE EXPERIMENTAL

Nesta seção serão apresentados os testes individuais para cada sensor e comprovar o seu funcionamento comparando-os com equipamentos já consolidados no mercado.

3.1 Teste do sensor de tensão

De posse de um multímetro na escala de tensão alternada foi inserido as pontas de prova para verificar qual a tensão da rede elétrica, e em seguida comparada com a tensão que o sensor de tensão AC ZMPTB. O resultado do teste pode ser visto na Figura 22.

Figura 22: Resultado do teste do sensor de tensão elétrica



Fonte: Autoria própria

Analisando a Figura 22b a tensão elétrica capturada pelo sensor está coerente com a do multímetro, Figura22a, o que comprova o seu bom funcionamento no sistema. Todavia, é necessário calcular o erro relativo nas medidas realizadas na Figura 21 para que possamos definir as medidas de erros em porcentagem.

Na equação 1, é demonstrado o erro absoluto das medidas na qual é subtraído o valor obtido do valor esperado. Em seguida na equação 2 é calculado o erro relativo, na qual para obter o resultado é dividido o erro absoluto pelo valor real da leitura.

Equação 1- Cálculo do erro absoluto

$$E_A = 224 - 222 = 2V \quad (1)$$

Equação 2 – Cálculo do erro relativo

$$E_r = \frac{2V}{224} = 0{,}009 \quad (2)$$

$$E_r = 0{,}009 * 100 = 0{,}9\%$$

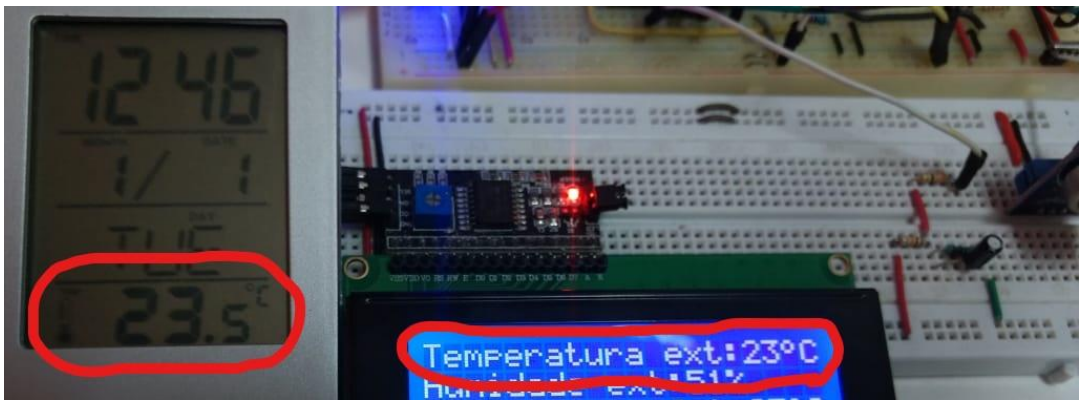
Após efetuar os cálculos, conclui-se que o erro relativo obtido na Equação 2 é de 0,9% o que é considerado um erro baixo.

3.2 Sensor de temperatura e humidade relativa do ar DHT22

Para o teste definitivo do sensor de temperatura e humidade relativa do ar DHT22 foi utilizado um sensor de temperatura de um relógio que contem a leitura da temperatura ambiente. Os testes podem ser vistos na Figura 23.

Em relação a humidade relativa do ar, os testes foram feitos a partir das leituras de estações meteorológicas compartilhadas na internet apenas para fins de comparação, todavia a humidade do ar na sala de servidores é diferente da humidade externa, pois o ar condicionado tem grande influência na diminuição da humidade.

Figura 23: Testes da temperatura ambiente



Fonte: Autoria própria

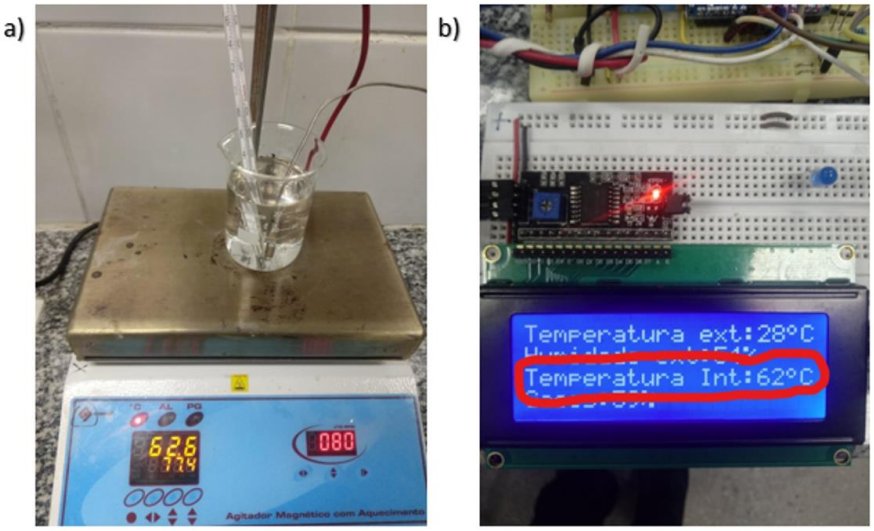
Como visto na Figura 23 a temperatura ext. (externa) está associada ao sensor de temperatura DHT22 e a sua precisão é bastante satisfatória. Utilizando as equações da seção 3.1, podemos verificar o erro relativo das leituras efetuadas na Figura 22. Visto isso o erro relativo encontrado foi de aproximadamente 0% visto que as leituras foram exatamente iguais para ambos os sensores.

3.3 Teste do sensor de temperatura termopar

Com o auxílio de um agitador magnético acoplado a um termômetro digital foi possível aquecer a água contida no *Becker* para que de forma controlada realizar o teste do termopar. A Figura 24 demonstra os procedimentos realizados.

Na Figura22a verificou-se que a leitura digital do termopar esta em torno de 62,6ºC e na Figura22b, é possível observar que a leitura do termopar também apresenta a mesma leitura, e com isso os testes se demonstraram eficientes. Para verificar a porcentagem de erro relativo das leituras realizadas nesta seção, pode-se utilizar as equações da seção 3.1, e com isso o erro relativo encontrado é de aproximadamente 0% visto que as leituras realizadas nos testes foram próximas.

Figura 24: Testes do sensor de temperatura (termopar)



Fonte: Autoria própria

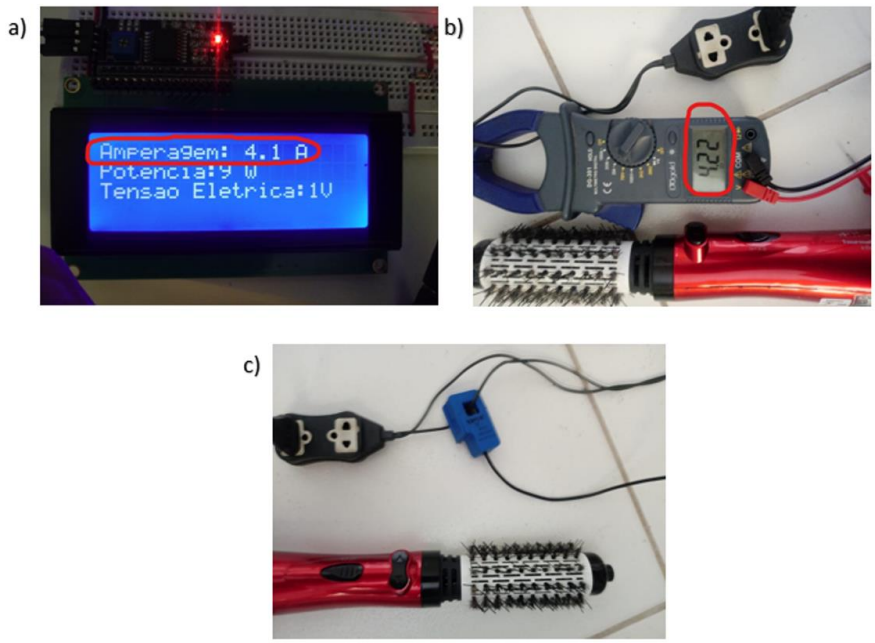
3.4 Teste do sensor de corrente elétrica

Com posse de um alicate amperímetro, da marca DG-301, verificou-se o consumo de corrente elétrica de um secador de cabelo, a corrente consumida do aparelho em sua potência máxima é de aproximadamente 4.22 A (amperes) como mostra a Figura25b. Na Figura25c, pode-se observar que a o sensor de corrente elétrica foi posicionado em apenas um dos fios que vem da rede elétrica na qual está ligado o secador, e na Figura a o *display* LCD mostra

que a o consumo de corrente é de 4.1A. Com isso pode-se comprovar que o funcionamento do sensor está dentro dos parâmetros.

Na Figura 25 pode ser visto os resultados das leituras entre o alicate amperímetro e o sensor de corrente 30A SCT-013. Ao realizar os testes, pode-se utilizar as equações da seção 3.1 para realizar os testes do erro relativo das medidas desta seção, e com isso conclui-se que o erro encontrado é de aproximadamente 4%, um erro significativamente pequeno e dentro do recomendado para esta aplicação.

Figura 25: Resultados das leituras do alicate amperímetro e do sensor de corrente



Fonte: Autoria própria

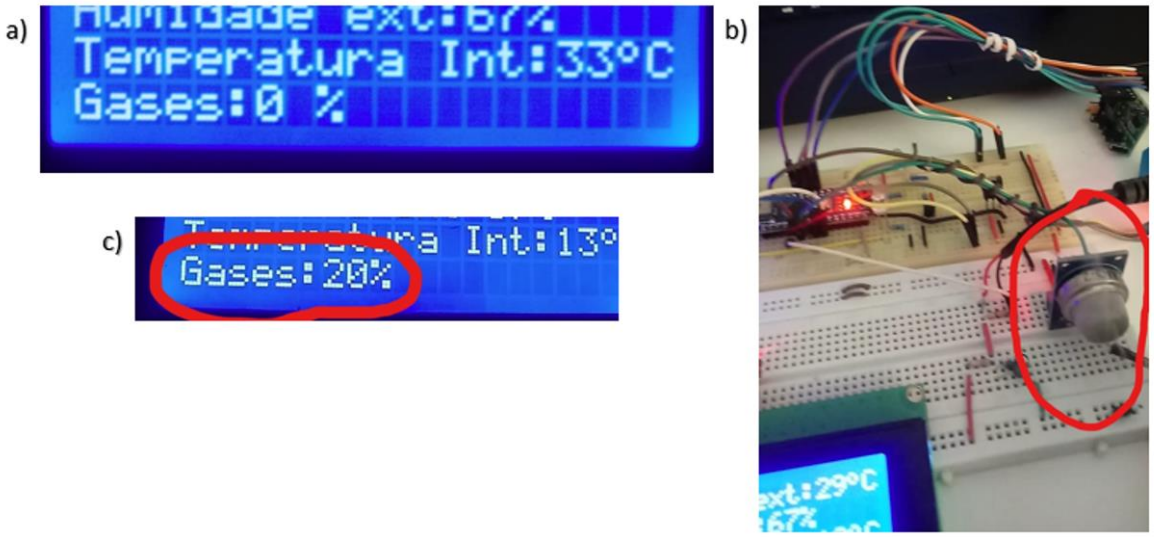
3.5 Teste do sensor de gases MQ-2

Para efetuar o teste do sensor de gases tóxicos MQ-2, vale salientar que neste projeto, a calibração do mesmo não é necessária, visto que o único objetivo é capturar leituras de gases que estejam dentro dos seus parâmetros de leituras, sem distinção de qual tipo de gás está no ambiente.

Após deixar o sensor ligado por pelo menos 3 horas, foi colocado fumaça de forma controlada no sensor para que possamos detectar a variação da tensão no seu pino analógico, que por sua vez será interpretado pelo microcontrolador do Arduino®, e em seguida capturar

o grau de porcentagem de gás no ambiente. Na Figura 26 pode ser visto os resultados das leituras do sensor de fumaça.

Figura 26: Resultados das leituras do sensor de fumaça MQ-2



Fonte: Autoria própria

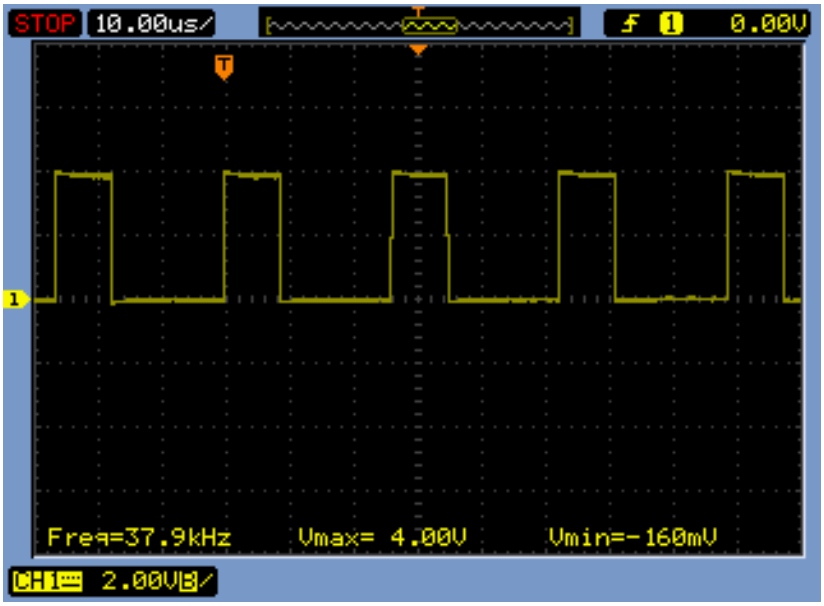
Perceba que na Figura26a o sensor está capturando 0% de concentração de ppm no ambiente, e após a calibração do aquecedor, foi colocado um palito de fósforo, conforme apresentado na Figura26b, para emitir fumaça e testar a sensibilidade do sensor, e com isso na Figura26c podemos perceber que o sensor detectou uma concentração de ppm no ambiente de 20%.

3.6 Teste do sensor infravermelho

Com posse de um osciloscópio, foi possível capturar a forma de onda que o Arduino® envia para o led infravermelho, essa onda tem características de uma onda quadrada com frequência de 38kHZ.

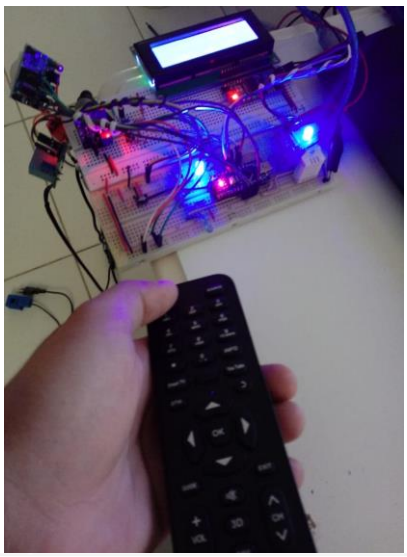
Para tanto, o comando enviado, vem a partir de um vetor de números em hexadecimal, cujo é interpretado pelo software e reproduzido em forma de níveis baixos e níveis alto. É possível ver a onda gerada pelo circuito infravermelho na Figura 27, e a captura do código hexadecimal na Figura 28.

Figura 27: Teste do sensor infravermelho



Fonte: Autoria própria

Figura 28: Captura do código do controle para reprodução no sensor infravermelho



```
{9050,50,200;4050,700,450,650,500,650,500,600,550,600,100,
150,250,650,500,600,500,650,1600,650,100,150,1300,650,100}
```

Fonte: Autoria própria

Uma observação importante é que para cada botão do controle a ser clonado, tem um código diferente, ou seja, se o objetivo fosse clonar mais funções além do controle da temperatura, o correto seria repetir os passos da Figura 28, cujo é capturar os códigos

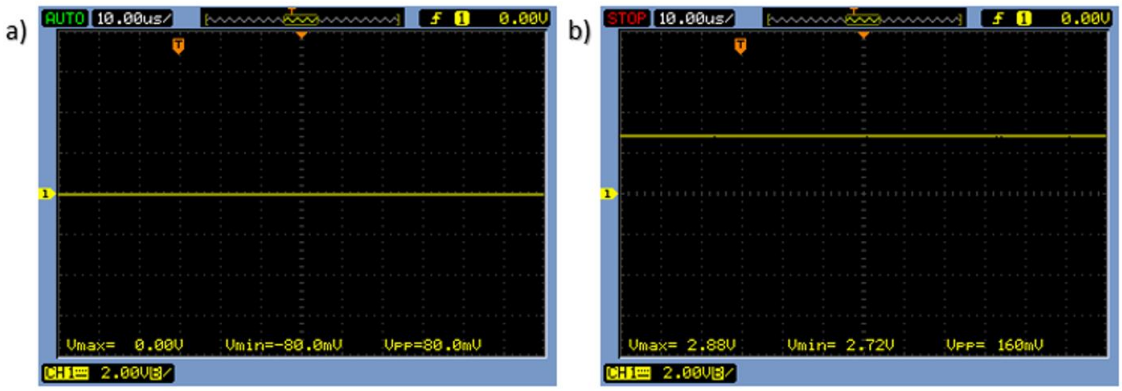
hexadecimal para cada botão individualmente, para que o Arduino® saiba exatamente qual comando enviar na hora de controlar o equipamento alvo.

3.7 Teste do sensor de presença PIR

Para o teste do sensor de presença, foi utilizado o mini osciloscópio para verificar a saída do pino digital do pino AUT (Output) do sensor, inicialmente foi analisado o pino quando o sensor não está detectando nenhum obstáculo, e foi observado que o nível lógico do pino era próximo de 0V, o que para o Arduino® é entendido como nível baixo. Em seguida, foi posto um obstáculo na frente do sensor PIR para verificar o comportamento da saída digital do mesmo, e que a saída foi para 3.28V, ou seja, o sensor passou a enviar nível alto para o Arduino®, e com isso o microprocessador interpreta essa informação e informa para o usuário que o sensor detectou movimento.

Com isso, conclui-se que o sensor de presença PIR se baseia em níveis lógicos 0 e 1 para detectar movimentação no ambiente em que o mesmo está instalado, essas saídas digitais podem ser calibradas para passar mais tempo em nível alto, ou menos tempo, assim como permite calibrar a sensibilidade de detecção, vai de acordo com a necessidade do usuário. Na Figura 29 pode ser visto os resultados.

Figura 29: Resultados dos testes no sensor de presença PIR



Fonte: Autoria própria

Perceba que na Figura29a o sensor está inicialmente com nível lógico baixo, ou seja 0, e após a detecção de movimentação, Figura29b mostra o sensor enviando nível alto para o Arduino® comprovando assim o seu funcionamento de forma eficiente para este projeto.

3.8 Análise dos resultados dos testes experimentais

Nesta seção 3.0 foi demonstrado o teste de cada sensor individualmente para comprovar o seu funcionamento, bem como realizando as devidas comparações com equipamentos profissionais para verificar se as leituras eram de fato compatíveis, com isso, após essa etapa, o Arduino® concatena todas as leituras dos sensores em uma única string e envia pela serial para a interface criada na linguagem Java, e na seção 2.13 pode ser visto a interface devidamente preenchida pelos dados, bem como as suas funcionalidades.

Pensando nisso, pode-se afirmar que o *software* se comporta como o esperado, porém não se pode deixar de comentar os possíveis erros cometidos no projeto, visto que o objetivo é diminui-los o máximo possível. Um dos erros a ser mencionado são os gráficos fornecidos pelo fabricante em relação ao sensor de gases MQ-2, pois os testes efetuados no datasheet dizem respeito a ambientes devidamente controlados, diferente da proposta deste trabalho, mas que não será de grande impacto para o objetivo do *software*.

Outros possíveis erros que também não podem deixar de ser mencionado, são os erros de interferência externa nos sensores, o que pode ocasionar o famoso alarme falso, ou uma leitura incorreta, por exemplo, o sensor receptor de infravermelho precisa está direcionado diretamente para o controle remoto no momento em que o mesmo realiza a captura do código de um determinado botão, se por ventura alguma interferência externa intervir na captura do código isso pode ocasionar a falta de alguns dados primordiais, fazendo com que o led emissor não consiga transmitir a informação para baixar ao temperatura do ar condicionado, ocasionando assim um envio de e-mail por alta temperatura.

E não menos importante, o erro humano também é um grande fator de risco para o sistema, pois manusear o *software* sem as devidas orientações podem levar a má interpretação dos dados, levando-o a uma possível paralisação do monitoramento, e consequentemente a sala de servidores ficará vulnerável a possíveis problemas que por ventura possam vir ocorrer. Assim podemos observar que mesmo nos testes individuais dos sensores e do *software* não foi possível garantir que houvesse a presença de variáveis indesejáveis, como por exemplo interferência externa.

4 CONCLUSÃO

Neste trabalho foi desenvolvido um sistema de monitoramento para sala de servidores, capaz de monitorar o ambiente como um todo. Utilizando a plataforma Arduino®, foi possível monitorar a concentração de gases tóxicos no ambiente, com o auxílio dos sensores da família MQ-X, mais especificadamente o MQ-2, o sistema também é capaz de monitorar a temperatura externa e interna e humidade relativa do ar com os sensores DHT22 e termopar. O sistema também monitora a tensão da rede elétrica bem como a sua corrente também para segurança da sala e das máquinas, a presença de pessoas na sala também é monitorada, bem como o controle automático do ar condicionado para manter a sala sempre na temperatura ideal, bem como o uso de LCD para redundância controlada de dados.

A metodologia utilizada para medição proposta neste trabalho foi o uso das folhas de dados dos fabricantes, bem como diversas pesquisas na internet sobre os mesmos. O código feito no Arduino® se comunica com a interface gráfica no feita na linguagem Java e com isso gerando assim gráficos, tabelas, e amostra de dados em tempo real para o usuário. Os erros encontrados durante o processo de testes dos sensores e da interface gráfica foram mínimos partindo de que neste trabalho trabalhar com objetivo de minimizar os erros foi cumprida.

Para trabalhos futuros é necessário o aprimoramento da interface gráfica, para que o usuário possa calibrar os sensores da forma como desejar, e desenvolver um novo experimento para o aprimoramento dos sensores para que com isso possamos diminuir ainda mais os erros.

Além do aprimoramento da interface como foi mencionado, o *software* pode se desenvolver para o monitoramento não somente de sala de servidores, mas para qualquer tipo de sala, e com isso poderá monitorar qualquer ambiente de forma genérica, e se adaptando conforme o usuário desejar, bem como uma interface web para monitoramento a longa distância.

Por fim, o sistema desenvolvido funciona bem em situações que necessite que uma determinada ação venha a ser necessária, na qual o monitoramento constante deve ser uma prioridade. Com isso este trabalho realiza bem os eu papel em monitorar uma sala de servidores de forma confiável. Porém, apesar das limitações, conclui-se que seja viável a construção de um sistema de monitoramento de baixo custo e programável.

REFERÊNCIAS

CANDIDO, Gradimilo. Sensor de Gás MQ-135 e a família MQ de detectores de Gás. 2017. Disponível em: <https://portal.vidadesilicio.com.br/sensor-de-gas-mq-135/>. Acesso em: 3 jan. 2020.
(CANDIDO, 2017)

MURTA, Gustavo. Sensores DHT11 e DHT22: Guia básico dos sensores de umidade e temperatura. Disponível em: <https://blog.eletrogate.com/sensores-dht11-dht22/>. Acesso em: 4 jan. 2020.
(MURTA, 2020)

COMSTOR, Canal. **O QUE É UM DATA CENTER?** 2013. Disponível em: <https://blogbrasil.comstor.com/bid/334188/o-que-um-data-center>. Acesso em: 11 out. 2019.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TECNICAS. ABNT NBR 14565; cabeamento estruturado para edifícios comerciais e data center. 4.ed. Rio de Janeiro; ABNT, 2013.

MACêDO, Diego. **Protegendo a sala de servidores e a rede.** 2017. Disponível em: <https://www.diegomacedo.com.br/protegendo-a-sala-de-servidores-e-a-rede/>. Acesso em: 11 out. 2019.

YAMANOUE, Takashi. **Monitoring Servers, With a Little Help from my Bots.** 2017. Disponível em: <https://dl.acm.org/citation.cfm?id=3123461>. Acesso em: 11 out. 2019.

M. P. David, and R. R. Schmidt, "Impact of ASHRAE environmental classes on data centers," *Fourteenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, Orlando, FL, 2014, pp. 1092-1099.

CORREIA, Stella. **Monitoramento do data center: o que você precisa saber sobre a importância da sala de controle.** 2018. Disponível em: <http://blog.wdcnet.com.br/monitoramento-do-data-center/>. Acesso em: 12 out. 2019.

PAESSLER. **Monitoramento de sala de servidor o PRTG proteja sua sala de servidor.** 2020. Disponível em: <https://www.br.paessler.com/server_room_monitoring>. Acesso em: 01 Não é um mês valido! 2020.

UOL Economia (www.economia.uol.com.br, acesso em 29 de jan. 2020, 9:46)

ELETRONICS, Hanwei. **MQ-2 Semiconductor Sensor for Combustible Gas.** 2020. Disponível em: <https://img.filipeflop.com/files/download/Datasheet_Sensor_Gas_MQ2.pdf>. Acesso em: 30 jan. 2020.

ELECTRONICS, Aosong. **Digital-output relative humidity & temperature sensor/module.** Disponível em: <https://datasheet4u.com/datasheet-pdf/Aosong/DHT22/pdf.php?id=792211>. Acesso em: 30 jan. 2020.

FILIPIFLOP. **Sensor de Movimento Presença PIR.** 2020. Disponível em: <https://www.filipeflop.com/produto/sensor-de-movimento-presenca-pir/>. Acesso em: 31 jan. 2020.

COMPHAUS. **Arduino Nano.** 2020. Disponível em: <http://comphaus.com.br/home/?s=Arduino+nano>. Acesso em: 31 jan. 2020.

ELETRÔNICA, Bau da. **Módulo Sensor de Temperatura MAX6675 + Termopar Tipo K.** 2020. Disponível em: <https://www.baudaeletronica.com.br/modulo-sensor-de-temperatura-max6675-termopar-tipo-k.html>. Acesso em: 31 jan. 2020.

MASTERWALKER. **Sensor de Tensão AC 0 a 250V Voltímetro ZMPT101B.** 2020. Disponível em: <https://www.masterwalkershop.com.br/sensor-de-tensao-ac-0-a-250vac-zmpt101b?utm_source=SmartHint&utm_campaign=SmartHint-Recs&utm_medium=MostPopular>. Acesso em: 31 jan. 2020.

FURUKAWA. **TIATIA--942 942 Telecommunications Telecommunications Infrastructure Infrastructure Standard for Data CentersStandard for Data Centers1Standard for Data CentersStandard for Data Centers.** 2005. Disponível em: <http://www.itbusinessday.com.br/2010/conteudo/sergio.pdf>. Acesso em: 2 fev. 2020.

APÊNDICE A – ADICIONANDO DLLs

Inicialmente, as duas DLLs mais importantes para que haja a comunicação serial entre o Arduino® e a aplicação feita na linguagem de programação Java é a rxtxParelled.dll e a rxtxserial.dd.

Após baixar as duas DLLs via *Google* basta copiar e colar ambas na pasta System32 do seu sistema operacional, e o mesmo procedimento para a pasta sysWOW64 também do seu sistema operacional, para que após realizar o código no Arduino® e no Java, as DLLs permitem que a comunicação serial seja bem sucedida.

APÊNDICE B - CÓDIGO DO ARDUINO®

Código do Arduino

```
/**
  Código para controle e leitura dos sensores do projeto de monitoramento;

  @Autor: Felipe Cardoso Pimenta.

  @Versão: 1.2.0.

  @Data setembro de 2019
*/

// -------- Bibliotecas auxiliares -------
#include <dht.h>                    //Biblioteca auxiliar para o sensor de temperatura e humidade do ar.
#include "EmonLib.h"                //Biblioteca auxiliar para o sensor de corrente elétrica.
#include <IRremote.h>               //Biblioteca auxiliar para o controle do sensor infravermelho e o de captura do código Hexadecimal.
#include <Wire.h>
#include <LiquidCrystal_I2C.h>      //Biblioteca auxiliar para o controle do display LCD com I2C
#include "max6675.h"                //Biblioteca auxiliar para o termopar.
```

```
// ---------------------------------------

// ---- Mapeamento de hardware e constantes auxiliares -------
#define DHT 10                    // Pino de leitura do sensor de humidade e temperatura do ar.
#define pinReceptor 11
#define pinLed 12
#define frequencia 38 // kHz
#define entradaAnalogicaMq2 A3 // constante que define o pino do sensor de gases.
#define tensaoDaRede 220
#define VOLT_CAL 458.1 //VALOR DE CALIBRAÇÃO (DEVE SER AJUSTADO EM PARALELO COM UM MULTÍMETRO)
#define pinPIR 8
long int tempoDelay = 0x00;
long int tempoDeMensagem = 0x05;
int count = 0x00;
IRrecv receptorIR(pinReceptor);
IRsend emissorIR;
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
//Código para gerar a bolinha do grau celsius no display LCD
byte grau[8] = { B00001100,
                 B00010010,
                 B00010010,
                 B00001100,
                 B00000000,
                 B00000000,
                 B00000000,
                 B00000000,
               };

//mapeamento de hardware para o termopar
int thermoDO = 5;
int thermoCS = 6;
int thermoCLK = 7;
// ---------------------------------------
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

int pinSCT = A0; // Variável auxiliar para leitura do sensor de corrente elétrica.
dht meuDHT;
EnergyMonitor emon1; // cria um objeto para o sensor de corrente.
EnergyMonitor emon2; // cria um objeto para o sensor de tensão.
// ---------------------------------------------------------

bool lerComando = false; // Inicia a variável lerComando inicialmente em false.

// ---------- Funções auxiliares ------------------------------------------
void ircode   (decode_results *results);
void encoding (decode_results *results);
void dumpCode (decode_results *results);
void acionareceptorInfravermelho();
```

```
void preencheDisplayLCD(int temperatura, int humidade, int leituraMQ2, double
corrente, int potencia, int temperaturaInterna);
void preencheDisplayLCD_2(double corrente, int potencia, int tensao);
// ---------------------------------------------------------------------

// Frequência capturada pelo sensor receptor infravermelho.
unsigned int teclaA[] = {9050, 4300, 700, 450, 650, 450, 700, 450, 650, 450, 700, 450, 650,
450, 700, 450, 650, 450, 700, 1550, 650, 1600, 650, 1600, 650, 1600, 650, 1550, 700, 1550,
650, 500, 650, 1550, 700, 450, 650, 1600, 700, 1550, 650, 450, 650, 1600, 700, 450, 650, 450,
650, 500, 650, 1550, 650, 500, 650, 500, 650, 1550, 650, 500, 650, 1600, 650, 1550, 700,
1550, 700};

// ---- Variáveis auxiliares ---------
String valores = "";
String presenca = "";
int temperatura = 0x00,
   humidade = 0x00, temperaturaInterna = 0x00;
int leituraMQ2 = 0x00;
double Corrente = 0x00;
int potencia;
int amostragem = 10;
long intervalo = 3000;
unsigned long currentMillis = millis();
int dispState = 1;
unsigned long previousTime = 3000;
int tensao;
float supplyVoltage;
// ----------------------------------

void setup() {
  Serial.begin(9600);                                  // Comunicação serial do árduino
configurado para 9600 bound rage;
  emon1.current(pinSCT, 60);                           // Iniciando as leituras do
sensor de corrente elétrica.
  pinMode(pinPIR, INPUT);                              // Definindo pino do sensor de
presença como saída digital.
  pinMode(pinLed, OUTPUT);
  lcd.backlight();
  lcd.begin(20, 4);                                    // Iniciando o display LCD.//
definindo o pino digital do led como sáida.
  lcd.setCursor(3, 0);
  lcd.print("SEJA BEM VINDO!");
  lcd.setCursor(3, 2);
  lcd.print("Felipe Cardoso");


  //Cria o caractere customizado com o simbolo do grau
  lcd.createChar(0, grau);
  // INICIANDO RECEPTOR IR
  receptorIR.enableIRIn();
```

```
  emon2.voltage(2, VOLT_CAL, 1.7); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGIO / VALOR DE CALIBRAÇÃO / MUDANÇA DE FASE)

}// end void setup.


void loop() {

  // -------------------- Leitura dos sensores do sistema de monitoramento ---------------------------------------------------------
  char leituraSerial = Serial.read(); //Variável que ira armazenar o comando enviado pela serial para realizar s leituras dos dados dos sensores.

  if (leituraSerial == 'l') {

    meuDHT.read22(DHT); // Realiza as leituras do pino digital 7.
    leituraMQ2 = analogRead(entradaAnalogicaMq2);     // Realizando a leitura do pino analógico do árduino A0.
    leituraMQ2 = map(leituraMQ2, 0, 1023, 0, 100 );
    int MQ2; // Variável para armazenar leitura do sensor de gás MQ-2 com condições no if e else.

    if(leituraMQ2 <= 5) {
      MQ2 = 0x00;
    } else {
      MQ2 = leituraMQ2;
    }
    temperatura = meuDHT.temperature;
    humidade = meuDHT.humidity;

    // Trecho de código responsável por obter a potência e a corrente elétrica.
    double Irms = emon1.calcIrms(1480); // Calcula o valor da corrente.
    potencia = Irms * tensaoDaRede; // Calcula o valor da potência Instantanea.

    if (Irms <= 0.18) {
      Corrente = 0x00;
      potencia = 0x00;
    } else {
      Corrente = Irms; // passando as leituras da corrente elétrica para uma variável global.
    }
    // ------------------------------------------------------------------------

    // Trecho de código responsável por obter a tensão
    emon2.calcVI(17, 2000); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS, TEMPO LIMITE PARA FAZER A MEDIÇÃO)
    supplyVoltage   = emon2.Vrms; //VARIÁVEL RECEBE O VALOR DE TENSÃO RMS OBTIDO
    tensao = supplyVoltage;
    // -------------------------------------------
```

```
  for (int index = 0; index < amostragem; index++) {
    temperaturaInterna = thermocouple.readCelsius() + temperaturaInterna;
  }// end laço for para leitura do termopar.

  // tira a média das leituras do sensor de temperatura (termopar).
  temperaturaInterna = temperaturaInterna / amostragem;

   // Trecho de código responsável por controle do sensor de presença
    bool valorPIR = digitalRead(pinPIR);
    if(valorPIR) {
      presenca = "DETECTADO";
    } else {
      presenca = "------";
    }
    // ------------------------------------------------------------

  //Concatena os dados lidos pelos sensores para enviar via serial, e serem tratados pela aplicação java.
  valores = String(temperatura) + ";" + String(humidade) + ";" + String(MQ2) + ";" + String(Corrente) + ";" + String(potencia) + ";" + String(temperaturaInterna) + ";" + String(tensao) + ";" + presenca; // Concatenando as informações obtidas para posteriormente enviar pela serial.

  //Trecho de código que faz a tansição de tela no display LCD
  if ((millis() - tempoDelay) >= tempoDeMensagem) {
    //Chamada da função que irá preencher os dados no display LCD.
    if (count == 0x00) {
      preencheDisplayLCD(temperatura, humidade, MQ2, Corrente, potencia, temperaturaInterna);
      count++;
    } else if (count == 0x01) {
      preencheDisplayLCD_2(Corrente, potencia, tensao);
      count = 0x00;
    }
    tempoDelay == millis();
  }
  // ------------------------------------------------------------
  Serial.println(valores); // Enviando os dados obtidos através das leituras para a serial do computador.

 } else if (leituraSerial == 't') {
  emissorIR.sendRaw(teclaA, sizeof(teclaA) / sizeof(teclaA[0]), frequencia);
 } else if (leituraSerial == 'b') {
  acionareceptorInfravermelho();
 }

}// end void loop.

void acionareceptorInfravermelho() {
```

```
  lerComando = true;
  digitalWrite(pinLed, HIGH);
  // LAÇO PARA LEITURA DO RECEPTOR IR QUANDO FOR PRESSIONADO O
BOTÃO
  while (lerComando) {
    decode_results  results;
    if (receptorIR.decode(&results)) {
      ircode   (&results);
      encoding (&results);
      dumpCode (&results);
      receptorIR.resume();
      lerComando = false;
      digitalWrite(pinLed, LOW);
    }
  }
}


void  ircode (decode_results *results)
{
  // Panasonic has an Address
  if (results->decode_type == PANASONIC) {
    Serial.print(results->address, HEX);
  }
}

void  encoding (decode_results *results)
{
  switch (results->decode_type) {
    default:
    case UNKNOWN:          break ;
    case NEC:             break ;
    case SONY:             break ;
    case RC5:             break ;
    case RC6:             break ;
    case DISH:             break ;
    case SHARP:            break ;
    case JVC:             break ;
    case SANYO:            break ;
    case MITSUBISHI:       break ;
    case SAMSUNG:          break ;
    case LG:              break ;
    case WHYNTER:          break ;
    case AIWA_RC_T501:     break ;
    case PANASONIC:        break ;
    case DENON:            break ;
  }
}

void  dumpCode (decode_results *results)
```

```
{
  for (int i = 1;  i < results->rawlen;  i++) {
    Serial.print(results->rawbuf[i] * USECPERTICK, DEC);
    if ( i < results->rawlen - 1 ) Serial.print(",");
    if (!(i & 1));
  }

  if (results->decode_type != UNKNOWN) {
    if (results->decode_type == PANASONIC) {
      Serial.print(results->address, HEX);
    }
    Serial.print(results->value, HEX);

  }
}

void preencheDisplayLCD(int temperatura, int humidade, int leituraMQ2, double
corrente, int potencia, int temperaturaInterna)
{
  // Trecho de código responsável por imprimir no LCD as leituras da temperatura
externa
  lcd.setCursor(0, 0);
  lcd.print("Temperatura ext: ");
  lcd.setCursor(16, 0);
  lcd.print(temperatura);
  lcd.setCursor(18, 0);
  lcd.write((byte)0);
  lcd.setCursor(19, 0);
  lcd.print("C");
  // ---------------------------------------------------------------------------

  // Trecho de código responsável por imprimir no LCD as leituras da humidade do
ambiente
  lcd.setCursor(0, 1);
  lcd.print("Humidade ext: ");
  lcd.setCursor(13, 1);
  lcd.print(humidade);
  lcd.setCursor(15, 1);
  lcd.print("%");
  // ---------------------------------------------------------------------------

  // Trecho de código responsável por imprimir no LCD as leituras da temperatura
interna
  lcd.setCursor(0, 2);
  lcd.print("Temperatura Int: ");
  lcd.setCursor(16, 2);
  lcd.print(temperaturaInterna);
  lcd.setCursor(18, 2);
  lcd.write((byte)0);
  lcd.setCursor(19, 2);
```

```
  lcd.print("C");
  // ---------------------------------------------------------------------------------

  // Trecho de código responsável por imprimir no LCD as leituras do sensor de gases
  lcd.setCursor(0, 3);
  lcd.print("Gases: ");
  lcd.setCursor(6, 3);
  lcd.print(leituraMQ2);
  lcd.setCursor(8, 3);
  lcd.print("%");
  dispState = 2;
  // ------------------------------------------------------------------------------------
}

void preencheDisplayLCD_2(double corrente, int potencia, int tensao) {

  lcd.clear();   // Limpando a tela do sisplay LCD
  // Trecho de código responsável por imprimir no LCD as leituras do sensor de corrente
elétrica.
  lcd.setCursor(0, 0);
  lcd.print("Amperagem: ");
  lcd.setCursor(11, 0);
  lcd.print(corrente);
  lcd.setCursor(14, 0);
  lcd.print(" A");
  // ------------------------------------------------------------------------------------

  // Trecho de código responsável por imprimir no LCD as leituras da potência do
equipamento.
  lcd.setCursor(0, 1);
  lcd.print("Potencia: ");
  lcd.setCursor(9, 1);
  lcd.print(potencia);
  lcd.setCursor(10, 1);
  lcd.print(" W");
  // ------------------------------------------------------------------------------------

  // Trecho de código responsável por imprimir no LCD as leituras do sensor de tensão
  lcd.setCursor(0, 2);
  lcd.print("Tensao Eletrica:");
  lcd.setCursor(16, 2);
  lcd.print(tensao);
  lcd.print("V");
  // ------------------------------------------------------------------------------------

}
```