PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

MONITORAMENTO DE TEMPERATURA, UMIDADE E GASES EM *DATA CENTERS*

BRUNO BARROS RIBEIRO

GOIÂNIA
2020

BRUNO BARROS RIBEIRO

MONITORAMENTO DE TEMPERATURA, UMIDADE E GASES EM *DATA CENTERS*

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientadora: Prof.ª Ms. Angélica da Silva Nunes

GOIÂNIA
2020

BRUNO BARROS RIBEIRO

MONITORAMENTO DE TEMPERATURA, UMIDADE E GASES EM *DATA CENTERS*

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em 03/12/2020.

_____

Prof.ª Ms. Ludmilla Reis Pinheiro dos Santos
Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientadora: Prof.ª Ms. Angélica da Silva Nunes

_____

Prof.º Ms. Rafael Leal Martins

_____

Prof.º Ms. Wilmar Oliveira de Queiroz

GOIÂNIA
2020

À minha mãe Ana Aparecida, e ao meu pai, Neidson pelo apoio, carinho e dedicação.

À minha tia, Silvia e meu tio, Dimas pelo apoio e a dedicação.

Aos meus colegas de faculdade, que sempre forneceram o apoio moral e instrucional.

AGRADECIMENTOS

A Professora Angélica Nunes, orientadora acadêmica, pelo suporte e dedicação na orientação deste trabalho.

Aos professores da Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás pelo conhecimento e colaboração.

“Não julgue cada dia pela colheita que você obtém, mas pelas sementes que você planta.”

Robert Louis Stevenson

RESUMO

Este trabalho de conclusão de curso apresenta um estudo teórico e prático sobre a segurança física dentro de um ambiente de *Data Center*. Mostrando primeiramente conceitos mais teóricos sobre os *Data Centers* e a importância de sua segurança e preservação do meio físico para o bom funcionamento dos equipamentos e a proteção de capital humano. Em seguida é documentado o passo a passo para a criação de uma central de monitoramento, que aborda etapas de construção de uma maquete que simula um ambiente físico um *Data Center* e trabalha a implementação de um conjunto de dispositivos e ferramentas que juntas criam um sistema capaz de monitorar as condições do ambiente físico do modelo representativo do *Data Center* de forma independente e remota, além de apresentar testes e resultados que validam a utilização da central monitoramento para a análise do ambiente e uso prático.

Palavras-Chaves: Central de monitoramento*, Data Centers,* Segurança Física.

ABSTRACT

This course conclusion work presents a theoretical and practical study of the physical security within a data center environment. First showing more theoretical concepts about Data Centers and the importance of their security and preservation of the physical environment for the proper functioning of equipment and the protection of human capital. Then, the step-by-step for the creation of a monitoring center is documented, which addresses the construction stages of a model that simulates a physical environment, a Data Center and implements a set of devices and tools that together create a system capable of monitoring the conditions of the physical environment of the representative model of the Data Center independently and remotely, in addition to presenting tests and results that validate the use of the central monitoring for the analysis of the environment and practical use.

Keywords*: Monitoring center, Data Centers, Physical Security.*

LISTA DE ILUSTRAÇÕES

Figura 1 - Modelo Representativo do *Data Center* 2D 28
Figura 2 - Modelo Representativo do *Data Center* 3D 29
Figura 3 - Dimensões da estrutura da maquete em tamanho real 30
Figura 4 - Peças em MDF após os cortes 31
Figura 5 - Maquete logo após a montagem 31
Figura 6 - Maquete finalizada 32
Figura 7 - Diagrama do fluxo de dados entre microcontrolador e servidores 33
Figura 8 - Módulo Wi-Fi ESP8266 NodeMcu ESP-12E 35
Figura 9 - Sensor de gás inflamável MQ-2 36
Figura 10 - Sensor DHT11 37
Figura 11 - Circuito elétrico 38
Figura 12 - Exemplo de estrutura utilizada na programação 40
Figura 13 - Bibliotecas 41
Figura 14 - Mapeamento do *Hardware* 41
Figura 15 - Declaração de variáveis e objetos 42
Figura 16 - Configuração inicial 43
Figura 17 - Estrutura de repetição, parte 1 44
Figura 18 - Estrutura de repetição, parte 2 45
Figura 19 - Estrutura de repetição, parte 3 45
Figura 20 - Simulação com sensor de temperatura e umidade (DHT11) 47
Figura 21 - Simulação do sensor de gás (MQ-2) 48
Figura 22 - Circuito elétrico dos testes 49
Figura 23 - Diagrama cliente-servidor 51
Figura 24 - Painel de controle do XAMPP (v3.2.4) 52
Figura 25 - Estrutura do banco de dados no PHPMyAdmin 53
Figura 26 - Interface PHPMyAdmin 54
Figura 27 - Diagrama de envio de dados através de requisições GET 55
Figura 28 - Código de envio da requisição GET, presente no microcontrolador 56
Figura 29 - Código do arquivo "connect.php", responsável pela conexão com banco de dados 57
Figura 30 - Código do arquivo "get.php", responsável pelo envio ao banco de dados 58
Figura 31 - *Interface* da plataforma *web* de monitoramento 59
Figura 32 - Arquivos da página *web* localizados no XAMPP 60
Figura 33 - Trecho de código em HTML da página de monitoramento 62
Figura 34 - Trecho de código utilizando a função Ajax, para atualização dos valores de temperatura 64
Figura 35 - Trecho em HTML, que recebe os valores atualizados da função Ajax 64
Figura 36 - Resulta da exibição do elemento na página de monitoramento 64
Figura 37 - Arquivo "db.class.php" utilizado para a conexão com o banco de dados 65
Figura 38 - Arquivo "temp.php" utilizado para realizar a consulta ao banco de dados 66
Figura 39 - Gráfico de temperatura dos últimos 5 minutos 67
Figura 40 - Gráfico de temperatura da semana 68
Figura 41 - Código utilizado para definir os parâmetros do gráfico de temperatura 69
Figura 42 - Função Ajax para atualização do gráfico de temperatura 70
Figura 43 - Trecho de código em HTML que recebe o gráfico já atualizado da função Ajax 70
Figura 44 - Estados de alerta de gás 71
Figura 45 – Tabela com valores de umidade, gás e temperatura 71

Figura 46 - Trecho de código em HTML utilizado para criação da tabela de umidade 72
Figura 47 - Diagrama com fluxo de funcionamento da central de monitoramento 73
Figura 48 - Compartimento onde se encontra o microcontrolado na maquete 74
Figura 49 - Processo de aquecimento com secador de cabelo 76
Figura 50 - Processo para liberação de gás butano com isqueiro 77
Figura 51 - Mudanças após testes com aquecimento de temperatura, gráfico de 5 minutos 78
Figura 52- Mudanças após testes com aquecimento de temperatura, gráfico da semana 78
Figura 53 - Alerta antes dos testes 79
Figura 54 - Alerta depois dos testes 79

LISTA DE TABELAS

Tabela 1 - Ameaças físicas dentro de um *Data Center* 23
Tabela 2 - Padrões de temperatura e umidade sugeridos com base no ASHRAE TC9.9 25

LISTA DE SIGLAS

2D	Duas Dimensões
3D	Três Dimensões
AJAX	*Asynchronous JavaScript and XML* - Javascript Assíncrono e XML
CPU	*Central Processing Unit* - Unidade Central de Processamento
CSS	*Cascading Style Sheets* - Folha de Estilo em Cascatas
GLP	Gás Liquefeito de Petróleo
GND	*Graduated Neutral Density* - Filtro Graduado de Densidade Neutra
GPIO	*General Purpose Input/Output* - Entrada e Saída de Uso Geral
GPU	*Graphics Processing Unit* - Unidade de Processamento Gráfico
HTML	*Hypertext Markup Language* - Linguagem de Marcação de Hipertexto
HTTP	*HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto
IDE	*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
LED	*Light Emitting Diode* - Diodo Emissor de Luz
MDF	*Medium Density Fiberboard* - Fibra de Média Densidade
NC	*Not Conected* - Não Conectado
NTC	*Negative Temperature Coefficient* - Coeficiente de Temperatura Negativo
PCB	*Print Circuit Board* - Placa de Circuito Impresso
PDU	*Power Distribution Unit* - Unidade de Distribuição de Energia
PHP	*Hypertext Preprocessor* - Pré-processador de Hipertexto
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language* - Linguagem de Consulta Estruturada
TI	Tecnologia da Informação
UPS	*Uninterruptible Power Supply* - Fonte de Energia Ininterrupta
URL	*Uniform Resource Locator* - Localizador Uniforme de Recursos
VCC	Voltagem em Corrente Contínua
WAN	*Wide Area Network* - Rede de Longa Distância
WEB	*World Wide Web* - Rede Mundial de Computadores
WI-FI	*Wireless Fidelity* - Fidelidade sem fio
XML	*eXtensible Markup Language* - Linguagem de Marcação Extensível

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO 15
1.1. Justificativa 16
1.2. Objetivo do Trabalho 16
1.3. Metodologia 16
1.4. Estrutura do Trabalho 17
CAPÍTULO 2 *DATA CENTERS* E SUA SEGURANÇA 18
2.1. Definição 18
2.2. Função 18
2.3. Infraestrutura 19
2.4. Dados 19
2.5. Segurança 20
2.5.1. Segurança Digital 21
2.5.2. Segurança Física 21
2.5.2.1. Controle de Acesso Físico 22
2.5.2.2. Circuito Fechado de Televisão (CFTV) 22
2.5.2.3. Controle do Ambiente Físico 22
2.5.3. Central de Monitoramento 25
CAPÍTULO 3 IMPLEMENTAÇÃO DE UMA CENTRAL DE MONITORAMENTO 27
3.1. Introdução 27
3.2. Modelo Representativo do *Data Center* 27
3.2.1. Modelo 2D (Duas Dimensões) 27
3.2.2. Modelo 3D (Três Dimensões) 29
3.2.3. Maquete 30
3.3. Construção do Projeto do Microcontrolador 32
3.3.1. Componentes 33
3.3.1.1. Microcontrolador Wi-Fi ESP8266 NodeMcu ESP-12E 34
3.3.1.2. Sensor de Gás MQ-2 35
3.3.1.3. Sensor de Temperatura e Umidade DHT11 36
3.3.2. Circuito Elétrico 37
3.3.3. Programação 39
3.3.3.1. Inclusão das Bibliotecas 40
3.3.3.2. Mapeamento do Hardware 41
3.3.3.3. Declaração de Variáveis e Objetos 42
3.3.3.4. Configuração Inicial 42
3.3.3.5. Estrutura de Repetição 44
3.3.4. Simulação 46
3.3.5. Circuito na Prática 48
3.4. Persistência dos Dados 49

3.4.1. Banco de Dados 50
3.4.2. Sistema Gerenciador de Banco de Dados 50
3.4.3. Criação do Servidor de Banco de Dados 51
3.4.3.1. Estrutura do Banco de Dados 53
3.4.3.2. Armazenamento dos Dados 54
3.4.3.3. Lado do Cliente 56
3.4.3.4. Lado do Servidor 57
3.5. Plataforma *Web* 58
3.5.1. Hospedagem no Servidor Web 59
3.5.2. Página de Monitoramento 61
3.5.3. Dados em Tempo Real 63
3.5.3.1. Exibição dos dados com Ajax 63
3.5.3.2. Exibição dos dados com PHP 65
3.5.4. Gráficos 66
3.5.5. Alerta de Gás e Fumaça 70
3.5.6. Tabelas de Consulta 71
CAPÍTULO 4 RESULTADOS E TESTES 73
4.1 Ambiente de Testes 74
4.1.1 Temperatura e Umidade 75
4.1.2 Gases e Fumaça 76
4.2 Resultados 77
4.2.1 Gráficos e Alarmes 80
4.2.2 Plataforma Web 80
4.2.3 Tempo de Resposta 80
CAPÍTULO 5 CONCLUSÃO 82
5.1 Sugestões de Trabalhos Futuros 83
BIBLIOGRAFIA 84

CAPÍTULO 1 INTRODUÇÃO

Entre 1990 e 2000, a *Internet* deixou de ser apenas de uso exclusivos de governos e universidades e passou também a ser de uso das massas. Em paralelo, havia uma grande demanda por tecnologia e computadores que pudessem auxiliar no crescimento de empresas que surgiam na época (REIS, 2019).

Com o aumento da quantidade de usuários e clientes, empresas focadas nas áreas de tecnologia da informação e comunicação, passaram por uma forte valorização dentro do mercado. Os negócios relacionados nesse meio recebiam cada vez mais capital de investimento com o intuito de formar uma Nova Economia nos EUA, que prometia ter fundos e crescimentos ilimitados, dando início a bolha da *Internet*. Assim, logo começaram a surgir diversas empresas na área movidas por esse impulso visando ganhar cada vez mais, de forma rápida e fácil (POZZEBOM, 2018).

Essa grande quantidade de empresas passou a precisar de uma rápida conexão de *Internet*, e que funcionasse de forma ininterrupta, para poder assim estabelecer suas marcas dentro do mercado tecnológico. Empresas essas, que prestam serviços como hospedagem de *websites*, gestão de *e-mails* e diversos serviços financeiros. Com isso, houve uma crescente necessidade de locais para manter os computadores-servidores da época, e infra estruturas que pudessem interligá-los e operá-los. Assim, alguns empreendedores tiveram a grande ideia de construir grandes galpões especializados em manter esses tipos de equipamentos, os chamados "*Data Centers*" (REIS, 2019).

Nos anos 2000, devido ao excesso de entusiasmo no setor, o *bug* do milênio, o aumento da taxa de juros americana e outros fatores, a bolha da *Internet* acabou estourando e muitas empresas faliram. Porém mesmo após a explosão da bolha, os chamados "*Data Centers*" se mantiveram até os dias de hoje, deixados como um legado muito importante da época (REIS, 2019).

Pode até não ser de conhecimento geral, mas atividades atuais como conversas *online*, *posts* em redes sociais, *e-mails*, pesquisas *online*, *sites* na *Internet* e qualquer outro serviço de "nuvem", tudo depende de servidores e equipamentos de redes para responder essas solicitações, e de infraestruturas que acomodam essa grande quantidade de equipamentos. Por isso é muito importante um bom projeto de desenvolvimento de um *Data Center* e de medidas de segurança adequadas para protegê-los, mantendo a integridade dos equipamentos e um constante funcionamento (SUMARES, 2016).

1.1. Justificativa

Os *Data Centers* são grandes centros de processamento e armazenamento de dados, onde são acomodados uma série de recursos computacionais e aplicações de empresas gigantescas nas mais diversas áreas da economia. Eles surgiram principalmente para proporcionar um melhor gerenciamento e proteção sobre os equipamentos e os dados ali instalados. Os equipamentos de um *Data Center* são extremamente caros e sensíveis, além disso a perda dos dados presentes nestes equipamentos pode levar as empresas e aos próprios clientes a terem prejuízos imensuráveis.

Por isso, é essencial que exista um sistema de monitoramento *web (World Wide Web* - Rede Mundial de Computadores*)*, que com o uso de sensores e alarmes ligados a um microcontrolador seja capaz de acompanhar em tempo real e remotamente estatísticas de temperatura, umidade e gases dentro do ambiente de um *Data Center*, proporcionando às empresas, uma ferramenta que permita a conservação climática do ambiente físico dos equipamentos, contribuindo para o seu bom funcionamento. Além disso, a ferramenta também proporciona a proteção de capital humano que possa estar realizando trabalho ou manutenção dentro do ambiente físico do *Data Center*.

1.2. Objetivo do Trabalho

O objetivo geral deste trabalho é proteger a integridade física do ambiente de um *Data Center*, contribuindo na segurança física do ambiente e na conservação do estado climático que ele se encontra.

O objetivo específico deste trabalho é criar um ambiente de teste simulado, uma maquete, que represente o ambiente físico de um *Data Center* em menor escala, e a implementação de um conjunto de dispositivos de monitoramento que funcione de forma autônoma no ambiente simulado pela maquete, e que utiliza-se de uma plataforma *web* para a visualização das condições climáticas que o ambiente se encontra em tempo real, podendo ser acessado de qualquer lugar remotamente e responsivamente.

1.3. Metodologia

Quanto a natureza deste trabalho, trata-se de um resumo de assunto, pois apenas trabalha e expande assuntos de trabalhos já publicados, que se tornaram a motivação para a criação deste

trabalho. Este resumo de assunto visa sistematizar uma área de conhecimento específico, mostrando sua evolução e sua importância.

Quanto aos seus objetivos, trata-se de uma pesquisa descritiva, pois com este trabalho busca-se obter dados mais consistentes sobre o ambiente físico em que os *Data Centers* se encontram, descrevendo os fatos como eles são. Tal pesquisa é feita através do levantamento de dados para análise.

Quanto aos seus procedimentos técnicos, trata-se de uma pesquisa experimental, caracterizada pela manipulação e obtenção de variáveis de um ambiente real, no caso, o ambiente físico simulado de um *Data Center*. Nesta pesquisa experimental, as variáveis de temperatura, umidade e gases são observadas, analisadas e controladas através de testes pelo pesquisador, e cuja medição nos leva a uma determinada conclusão através da análise dos resultados estatísticos aceitáveis e generalizáveis

1.4. Estrutura do Trabalho

O primeiro capítulo deste trabalho é a introdução, onde é falado um pouco sobre o surgimento dos *Data Centers* e a importância de sua segurança física, além das motivações, objetivos e os métodos deste trabalho de monografia.

O segundo capítulo aborda conceitos mais teóricos sobre os *Data Centers* e sua segurança, abordando também formas de corrigir falhas na segurança física e formas de monitoramento do ambiente.

No terceiro capítulo é abordado os passos de implementação de uma central de monitoramento, documentando cada parte, desde o planejamento até a construção, tanto do local de testes, do dispositivo que realizará o monitoramento, do servidor onde serão armazenados os dados coletados e da plataforma *web* de monitoramento.

No quarto capítulo são feitos testes para validar o funcionamento de todo o sistema de monitoramento, além da análise dos resultados gerados.

O quinto e último capítulo são descritas as conclusões obtidas através do desenvolvimento do trabalho prático e teórico, além das sugestões de trabalhos futuros que possam agregar mais valor ao que já foi desenvolvido.

CAPÍTULO 2 *DATA CENTERS* E SUA SEGURANÇA

2.1. Definição

Os *Data Centers* são grandes centros de armazenamento e processamento de dados, projetados por organizações de médio a grande porte que precisam armazenar e processar uma grande quantidade de dados para oferecer algum tipo de serviço sobre esses dados. Qualquer organização ou entidade que gera ou usa uma grande quantidade de dados tem a necessidade de manter esse tipo de serviço em algum nível, isso inclui órgãos públicos e educacionais, empresas de telecomunicações e varejo, instituições financeiras e empresas que oferecem serviços *online* e redes sociais como Google e Facebook (JOHNSON, 2020).

Toda mídia precisa ser armazenada em algum lugar. Assim, devido à grande quantidade, essas mídias estão migrando para a nuvem, o que significa que, em vez de armazená-las ou processá-las em nossos próprios computadores domésticos ou comerciais, é preferível acessá-las através de servidores de hospedagem de provedores de nuvem. A nuvem não significa que as aplicações e os dados não estejam localizados em *hardware*s, mas sim que alguém mantém esses dados em *hardware*s e *softwares* em locais remotos onde os clientes possam acessá-los via *Internet* , sem necessariamente saber o local físico de armazenamento, locais esses chamados de *Data Centers* (GRIFFITH, 2020).

Pode-se estimar que atualmente tenha mais de 7 milhões de *Data Centers* em todo mundo. Praticamente todo negócio e entidade governamental constroem e mantêm seus próprios *Data Centers* ou utilizam *Data Center* de outras empresas especializadas em manter esses tipos de serviços para empresas menores. Muitas opções estão disponíveis hoje, como alugar servidores em instalações compartilhadas, usar serviços de *Data Centers* gerenciados por terceiros ou usar serviços públicos baseados em nuvem de hosts como Amazon, Microsoft, Sony e Google (PALOALTO, 2020).

2.2. Função

No mundo corporativo da TI (Tecnologia da Informação), os *Data Centers* oferecem diversos tipos de serviços essenciais para os negócios, isso inclui (PALOALTO, 2020):

1. Armazenamento, gerenciamento, backup e recuperação de dados;
2. *Softwares* de produtividade, como *e-mail*;
3. Transações eletrônicas em grande volume;

4. Hospedagem de jogos *online*;
5. Hospedagem de *sites*;
6. Hospedagem de aplicações;
7. Serviços de *Big Data*, aprendizagem de máquina e inteligência artificial;
8. Máquinas virtuais e serviços de comunicação.

Porém, mais do que somente armazenar e processar dados, os *Data Centers* precisam garantir um funcionamento ininterrupto, com disponibilidade total, além de garantir a segurança e a confiabilidade dos dados ali armazenados (EVEO, 2019).

2.3. Infraestrutura

O projeto de um *Data Center* é baseado em uma rede de recursos computacionais e de armazenamento que permitem a entrega e o compartilhamento de aplicações e seus dados. Os principais componentes de um *Data Center* incluem roteadores, *switches*, *firewalls*, bancos de dados, servidores e controladores de aplicações, geralmente organizado em grandes *racks* e armários, localizados em grandes salas, galpões ou containers (CISCO, 2020).

Esses ambientes são muito bem planejados, quando possível, opta-se por locais menos suscetíveis a problemas geológicos, como terremotos, ou eventos climáticos, como chuvas fortes e furacões, sendo assim, locais livres de qualquer tipo de interferência natural (SUMARES, 2016).

Para que um *Data Center* funcione de forma adequada deve contar com uma infraestrutura de qualidade, com pisos elevados e uma estrutura alta para melhorar o fluxo de ar e acomodar uma grande quantidade de cabos, além de armários e *racks* muito bem organizados (EVEO, 2019).

É importante também, que possua uma estrutura capaz de abastecer energeticamente todos os equipamentos ali presentes de forma constante e sem oscilações, bem como um sistema que se garanta em caso de falhas. Assim, é comum esses locais apresentarem redes elétricas redundantes, contando com geradores, baterias, *nobreaks* e todo tipo de tecnologia capaz de garantir o funcionamento ininterrupto (ROSSI, 2013).

2.4. Dados

Os dados presentes dentro dos *Data Centers* variam desde informações levantadas pela própria organização, como, relação de vendas, estoque de produtos, esquemas e plantas de infraestruturas energéticas, sistemas de tráfego urbano até informações pessoais de clientes, como, documentos, contas bancárias, arquivos *e-mails* e outros.

É por isso que esses dados são de extrema importância, é muito importante que as empresas adotem medidas de segurança adequadas para protegê-lo. A falta de segurança pode acarretar a violação ou na perda dos dados, o que leva a exposição ou roubo das informações confidenciais da empresa, ou até pior, informações de clientes. Essa violação ou perda de informação pode ter um preço muito alto para as finanças da empresa e principalmente na sua reputação, e é por isso que existe um investimento pesado em segurança nesse setor, de forma a produzir barreiras impenetráveis aos dados (DOBRAN, 2019).

2.5. Segurança

A segurança de um *Data Center* envolve todo tipo de prática e preparação que mantenha os equipamentos, os dados e vida ali protegidos. Em um *Data Center* há vários tipos de ameaças, desde ataques cibernéticos e acessos não autorizados até superaquecimento e incêndios no ambiente físico.

Os serviços de segurança dos *Data Centers* é uma parte crítica de proteção da empresa, por isso, é essencial montar uma combinação certa de ferramentas e soluções de segurança. Por ser extraordinariamente difícil, ela abrange uma ampla gama de problemas, desafios e ameaças. O gerenciamento da infraestrutura do *Data Center* exige foco em diversas áreas, como gerenciamento remoto do ambiente, operações e procedimentos de segurança, infraestruturas de TI, engenharia de projetos, suporte a desktops virtuais e segurança cibernética (GREENGARD, 2019).

Devido à complexidade dos *Data Centers*, para protegê-los, os componentes de segurança devem ser analisados de forma separada, porém ao mesmo tempo com políticas de segurança bem similares e que se interligam. A segurança pode ser dividida em segurança digital, e segurança física. A segurança digital irá proteger os dados contra-ataques cibernéticos para evitar o acesso indevido e o vazamento de dados sigilosos. Por outro lado, a segurança física abrange uma ampla gama de processos e estratégias para conservar o bom funcionamento dos equipamentos e o ambiente que o cerca (FELTER, 2019).

2.5.1. Segurança Digital

Com a grande importância dos dados, os *Data Centers* precisam garantir a sua proteção contra ataques externos, via acessos remotos ou *softwares* maliciosos. Assim, *softwares* de segurança muito poderosos são utilizados para detectar e combater todo tipo de ameaça digital, como ferramentas de criptografia, autenticadores de usuários, controle de acessos remotos sofisticados, detectores de intrusões, *firewalls* e sistemas de prevenção (STALLINGS, 2015).

Softwares de segurança, além de evitar que usuários não autorizados acessem ou roubem dados confidenciais, eles também são usados para criar redundância, realizando *backups* a todo momento, protegendo o sistema contra perdas.

Todo ambiente de computação é diferente, é importante encontrar e implementar serviços de segurança apropriado para cada tipo de cenário. Isso muitas vezes requer um entendimento da topografia da rede, onde os dados são armazenados e como eles fluem pelo sistema, e a forma que as nuvens e outros sistemas afetam a segurança das aplicações e seus dados. Por isso, é necessário uma revisão completa e análise dos sistemas existentes e uma compreensão de onde ocorrem lacunas e falhas na segurança dos dados. Somente então é possível atender os requisitos de segurança do *Data Center* de maneira correta e eficaz. Para isso, existem diversas empresas, como: Cisco, Fortinet, IBM, Symantec, VMWare, FireEye e outras, especializadas em manter os dados protegidos em grandes centros de dados como os *Data Centers*, que oferecem serviços e ferramentas apropriadas para a proteção dos dados contra invasores externos (GREENGARD, 2019).

2.5.2. Segurança Física

De forma semelhante a segurança digital, a segurança física também visa proteger os dados, porém no seu âmbito físico. De forma a proteger o ambiente de alguma invasão interna e conservando um estado climático favorável para integridade dos equipamentos ali presentes, além da preservação de vidas humanas no local (FELTER, 2019).

A segurança física de um *Data Center* depende do seu porte e da sua localização. Mas o que eles têm em comum é a sua complexidade, geralmente são projetados desde o início com as melhores práticas de segurança em mente. Cada elemento ali presente deve ser implementado com outros elementos em mente, para assim, formar uma rede interligada de medidas de segurança que conversam e trabalham em conjunto (FELTER, 2019).

A segurança física é muito importante dentro de um *Data Center*, ela abrange uma ampla gama de processos e estratégias que são usados para proteger os equipamentos, os dados e a vida humana presentes neste tipo de ambiente.

2.5.2.1. Controle de Acesso Físico

Pode parecer uma coisa simples, mas uma das principais etapas da segurança física é a garantia de que apenas pessoas autorizadas acessem o local. Para isso, é importante que o local seja de difícil acesso e isolado das demais áreas de menor importância da empresa, e que todas as entradas tenham um método de autenticação automática, com detecção biométrica ou facial, além de cartões de acesso somente para o pessoal autorizado. O pessoal do local, incluindo guardas, pessoal de limpeza, manutenção e pessoal do centro de dados, deve submeter-se a verificações de antecedentes e devem ser acompanhados e aprovados pela segurança do *Data Center* para a entrada no local. Além disso, todo o pessoal deve ser orientado a observar intrusos ou atividades suspeitas (FELTER, 2019).

Controlar a movimentação de visitantes e funcionários no *Data Center* é crucial, é muito importante que tenha registros de quem teve acesso e quando, isso ajudará a investigar qualquer possível violação. Além disso, escadas de incêndio e rotas de evacuação deve permitir apenas que as pessoas saiam do local, não o contrário. Não deve haver acesso externo que permita a reentrada. E em casos isolados, havendo a entrada indevida, alarmes devem ser acionados (DOBRAN, 2019).

2.5.2.2. Circuito Fechado de Televisão (CFTV)

Todo tipo de registro é essencial, tanto entradas como saídas devem ser monitoradas através de câmeras de vigilância 24 horas por dia, 7 dias por semana. Registros esses que precisam ser mantidos por no mínimo três meses. Infelizmente algumas violações só são identificadas quando já é tarde demais, entretanto com esses registros podemos identificar falhas, indivíduos e vulnerabilidades (DOBRAN, 2019).

2.5.2.3. Controle do Ambiente Físico

Uma das principais partes da segurança física é o controle do ambiente, é preciso ter um controle do ar que entra e sai do local, assim como a temperatura e umidade em que o ambiente

se encontra para que os equipamentos funcionem normalmente sem nenhum tipo de superaquecimento ou corrosão, mantendo sua integridade e bom funcionamento (SUMARES, 2016).

Quando se fala em uma grande concentração de equipamentos eletrônicos, o calor e altas temperaturas são considerados indesejáveis. Grande parte dos componentes produzem calor a todo momento, e precisa se livrar dele rapidamente, para que o seu desempenho não caia ou para que seus componentes não se danifiquem. As CPUs (*Central Processing Unit* – Unidade Central de Processamento), por exemplo, podem fritar com seu próprio calor, assim como as GPUs (*Graphics Processing Unit* - Unidade de Processamento Gráfico), existe também janelas de temperaturas ideias para o funcionamento de discos rígidos. O resfriamento é realmente muito importante para os equipamentos do *Data Center*, principalmente por sua grande escala (HANAK, 2015).

Existem vários tipos de ameaças físicas em um ambiente como o *Data Center*, a Tabela 1 ilustra as principais ameaças existentes, o impacto que cada uma pode causar e os tipos de sensores utilizados para a detecção dela.

Tabela 1 - Ameaças físicas dentro de um *Data Center*

Ameaça	**Definição**	**Impacto**	**Sensores**
Temperatura do ar	Ar da sala, *rack* e equipamento de temperatura	Falha no equipamento e vida útil reduzida do equipamento em relação à temperatura acima da especificação e/ou mudanças drásticas de temperatura	Sensores de temperatura
Umidade	Umidade relativa do ambiente e do *rack* em temperatura específica	Falha no equipamento devido ao acúmulo de eletricidade estática em pontos de baixa umidade Formação de condensação em pontos de alta umidade	Sensores de umidade
Vazamentos de líquidos	Vazamentos de água ou refrigerante	Danos nos equipamentos Indicação de problemas no sistema de refrigeração ou na tubulação	Sensores de detecção de líquido

Ameaça	Definição	Impacto	Sensores
Erro humano e acesso de pessoal	Injustiça não intencional por pessoal Entrada não autorizada e/ou forçada no *Data Center* com intenção maliciosa	Danos ao equipamento e perda de dados Tempo de inatividade do equipamento Roubo e sabotagem de equipamentos	Câmeras de vigilância Sensores de movimento Contatos na porta Sensores de quebra de vidro Sensores de vibração
Fumaça / Fogo	Incêndio elétrico ou material	Falha no equipamento Perda de ativos e Dados	Sensores de fumaça e gases
Contaminantes perigosos transportados pelo ar	Produtos químicos no ar, como hidrogênio, provenientes de baterias e partículas como poeira	Situação perigosa para os funcionários e indício de falha do pessoal do UPS (*Uninterruptible Power Supply* - Fonte de Energia Ininterrupta) devido à liberação de hidrogênio Falha no equipamento devido ao aumento da eletricidade estática e entupimento dos filtros ou ventiladores devido ao acúmulo de poeira	Sensores químicos ou de hidrogênio Sensores de poeira

Fonte: COWAN; GASKINS, 2011.

Para cada sensor, condições operacionais aceitáveis devem ser pré-estabelecidas e configuradas para produzir alarmes quando as leituras excederem essas condições. Idealmente falando, o sistema que irá monitorar o *Data Center* deve ser flexível para configurar vários limites por sensor, com níveis de alerta diferentes para cada condição, como alerta crítico ou alerta de instabilidade. Além de limites de valor único, é importante que tenha condições de acionamento, como excesso de limite por um período de tempo, taxa de aumento e taxa de redução. No caso de medições na temperatura, alertar sobre taxas de alteração pode fornecer uma indicação mais rápida de falha do que somente capturar o alerta de uma temperatura única específica (COWAN; GASKINS, 2011).

Os limites e os valores ideais de funcionamento devem ser globalmente definidos como valores padrões e, em seguida, ajustados de acordo com a especificação da fabricante de cada

equipamento e o local de montagem do sensor em relação ao local do equipamento (por exemplo, um sensor próximo a uma fonte de alimentação de um servidor deve alarmar com um valor mais alto que um outro sensor localizado na entrada de ar do mesmo equipamento). Na Tabela 2, pode ser observado os limites padrões sugeridos para temperaturas e umidades para *Data Centers*, com base no ASHRAE TC9.9 (normas recomendadas pela Sociedade Americana de Engenheiros de Aquecimento, Refrigeração e Ar Condicionado, para ambientes de classe 1, que são mais rígidas e indicadas para ambientes como *Data Centers*). Além desses limites, é importante monitorar as taxas de mudanças de temperatura em certos períodos. Por exemplo, uma mudança de 5,6° C em um período de 5 minutos é uma provável indicação de falha do sistema de refrigeração do ambiente (COWAN; GASKINS, 2011).

Tabela 2 - Padrões de temperatura e umidade sugeridos com base no ASHRAE TC9.9

Sensor	**Valor máximo**	**Valor mínimo**
Temperatura	77°F (25°C)	68°F (20°C)
Umidade	55% de umidade relativa	40% de umidade relativa

Fonte: COWAN; GASKINS, 2011.

Todo *Data Center* também deve ser capaz de detectar gases combustíveis e fumaça no ar, para agilizar em um possível combate contra incêndio. Um sistema de prevenção contra incêndio é vital para que não haja risco de perda de vidas e nem de equipamentos caros. Esses sistemas são compostos por sensores de detecção de fumaça, extintores, gases inibidores e até brigadas de incêndio (RIBEIRO, 2013).

2.5.3. Central de Monitoramento

Para que os *Data Centers* possam operar de forma constante e sem problemas, é preciso que o ambiente se mantenha favorável o tempo inteiro. Todo fator externo que possa vir interferir no bom funcionamento do *Data Center* deve ser detectado em tempo real e comunicado às partes responsáveis para evitar qualquer tipo de infortúnio (HANAK, 2015).

Com o grande número de dispositivos que exigem atenção, avaliação de desempenho, análise comportamental e tomada de decisões, fica claro que é necessária uma plataforma que permita a visualização unificada de todos os subsistemas em um regime de 24/7, plataforma essa, chamada de central de monitoramento ou ferramenta de monitoramento (ROSA, 2018).

Em locais remotos, de difícil acesso, como filiais do interior ou centros de distribuição com acesso restrito, é inviável e pouco confiável ter pessoas se deslocando fisicamente para

realizar verificações no ambiente. Com a introdução da central de monitoramento, com acesso remoto via *Internet* fica muito mais barato para a organização realizar certos controles (COWAN; GASKINS, 2011).

A central de monitoramento, através de sensores e alarmes consegue acompanhar em tempo real e remotamente o que está acontecendo dentro do *Data Center*, sendo capaz de produzir estatísticas para a análise das condições em que o ambiente se encontra, fornecendo assim suporte necessário para que um analista monitore o *Data Center* e acione em caso de falhas cada setor responsável a tempo para realizar a correção necessária, tornando assim o ambiente favorável para o bom funcionamento dos equipamentos e para a conservação de vidas humanas que podem estar trabalhando no local ou realizando alguma manutenção (HANAK, 2015).

CAPÍTULO 3 IMPLEMENTAÇÃO DE UMA CENTRAL DE MONITORAMENTO

3.1. Introdução

Este capítulo visa documentar a implementação de uma central de monitoramento em um ambiente de teste simulando um *Data Center*, cujo objetivo é monitorar a temperatura, umidade e gases inflamáveis provenientes de incêndios. Tudo isso em tempo real, e com visualização remota através de uma plataforma *web*.

A criação da central de monitoramento do *Data Center* é dividida em quatro partes:

1. Criação de um Modelo Representativo de um *Data Center* (maquete): ambiente de testes onde os sensores serão posicionados e farão as medições necessárias no ambiente físico;
2. Implementação de um Projeto de Microcontrolador: responsável pela coleta e gerenciamento das medições, sendo um intermediador entre os valores capturados pelos sensores até o seu armazenamento em um banco de dados;
3. Criação de um Bancos de Dados: responsável pela persistência dos dados, necessário para a exibição na plataforma *web* de monitoramento;
4. Criação de uma Plataforma *Web*: utilizada para a visualização remota dos dados obtidos e alertas, recebidos em tempo real pelo banco de dados.

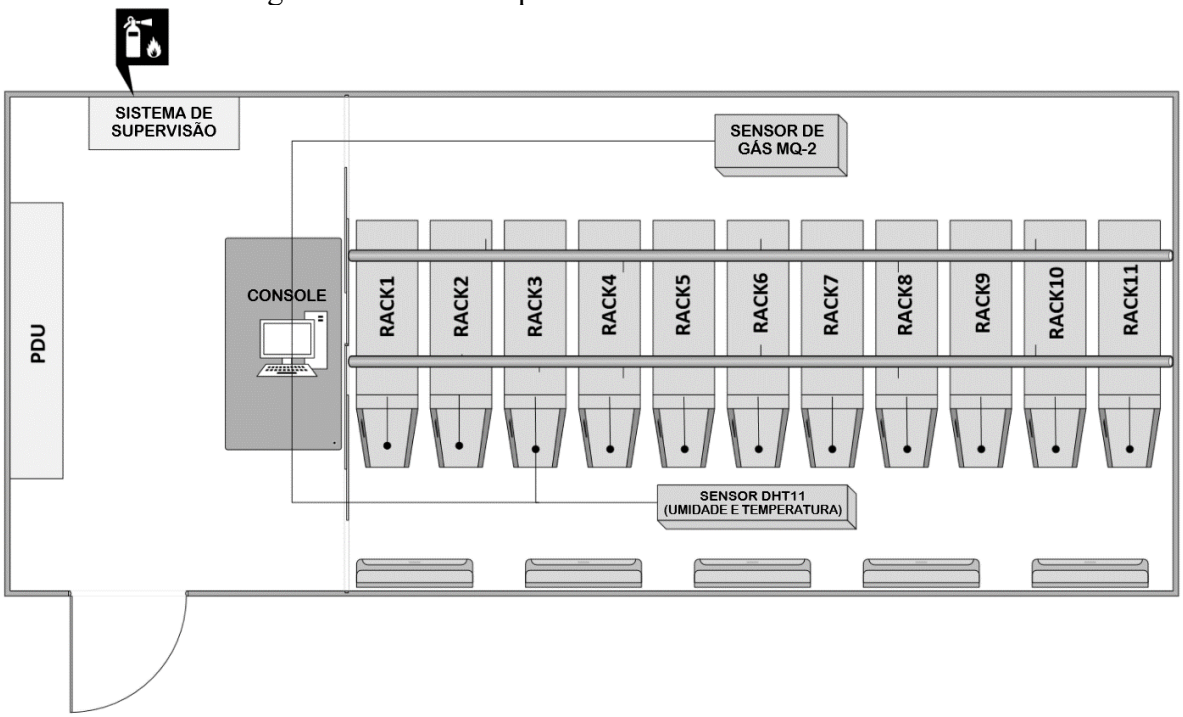
3.2. Modelo Representativo do *Data Center*

Esta seção visa trabalhar as etapas de criação de um modelo representativo de um *Data Center*, uma maquete, onde posteriormente são feitos os testes de monitoramento de temperatura, umidade e gases. Buscando mudanças que representam um ambiente real e situações anormais.

3.2.1. Modelo 2D (Duas Dimensões)

A primeira etapa consiste na elaboração de um modelo em 2D, com a estrutura básica a ser seguida de um *Data Center* real. Para isso, foi utilizado um *software* chamado Visio Standard 2019, disponibilizado pela Microsoft para criação de modelos.

Figura 1 - Modelo Representativo do *Data Center* 2D



Fonte: Elaborado pelo autor.

Na Figura 1, é possível observar o posicionamento dos diversos elementos importantes presentes em um *Data Center*. Cada um desempenha uma função específica:

- PDU (*Power Distribution Unit* - Unidade de Distribuição de Energia): é responsável por distribuir a energia elétrica para os equipamentos localizados no *Data Center*;
- Sistema de Supervisão: sistema de acesso local, utilizado para controlar a temperatura do ambiente, além de ser o responsável no combate contra incêndios com extintores e jatos de água em casos extremos;
- Console: computador de acesso local, utilizado para fazer alguma configuração nos equipamentos ou servidores;
- *Racks*: local onde ficam instalados os equipamentos de rede e servidores, bem semelhante a um armário;
- Dispositivos e Cabeamento: local do cabeamento estruturado elevado dos equipamentos dos *Racks*;
- Sensor de gás MQ-2: sensor de detecção de gases e fumaça;
- Sensor DHT11: sensor de temperatura e umidade.

Cada sensor é posicionado estrategicamente em um local que permita uma melhor avaliação física do ambiente. Os sensores de umidade e temperatura são posicionados na parte

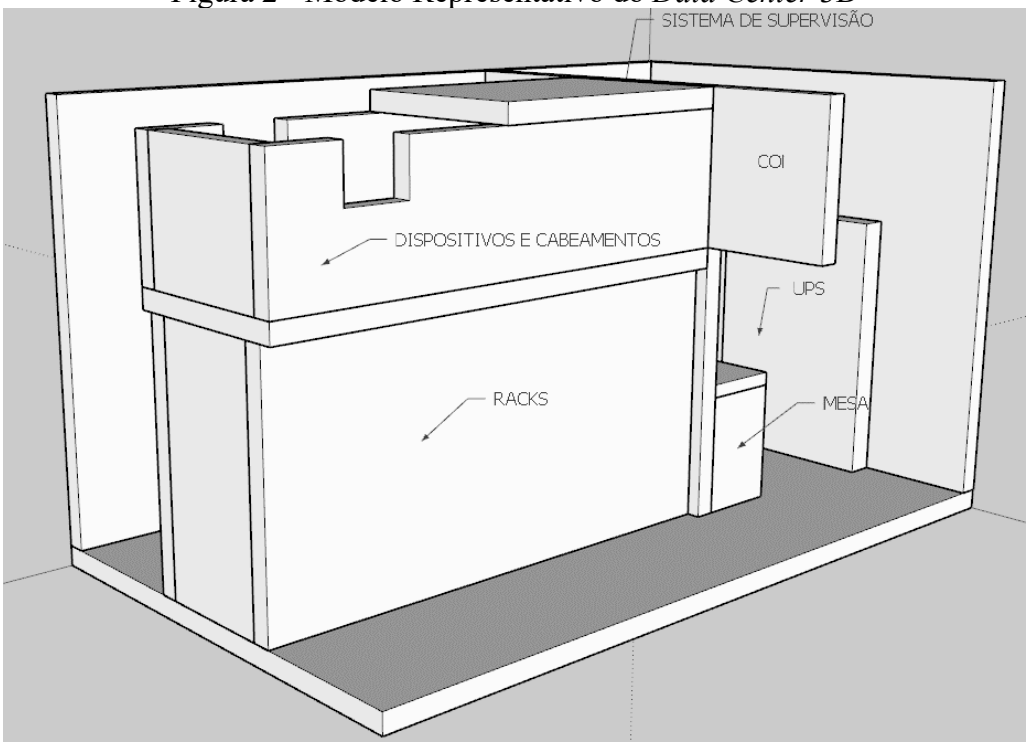
frontal dos *Racks*, onde seria a frente dos equipamentos instalados em um *Data Center* real, e o sensor de gás e fumaça se encontrará na parte traseira dos equipamentos, local mais suscetível a incêndios causados pelas altas temperaturas e curtos-circuitos.

3.2.2. Modelo 3D (Três Dimensões)

A próxima etapa consiste na criação de modelo 3D, representando o *Data Center* do primeiro modelo 2D. Porém o modelo 3D permite obter um melhor entendimento da estrutura do *Data Center*, assim como uma representação com dimensões em escala real para a confecção das peças e posterior montagem da maquete.

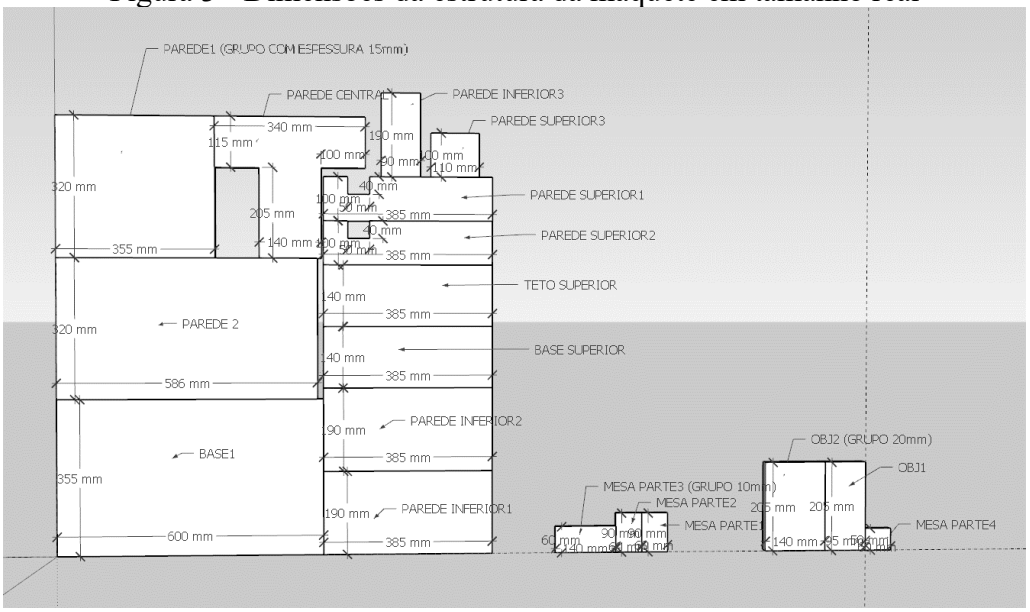
A elaboração do modelo em 3D foi feita utilizando um *software* chamado SketchUp, que é próprio para criação de modelos deste tipo, além de móveis e interiores. Extremamente prático o *software* permite obter uma projeção para as dimensões da estrutura da maquete em tamanho real, como mostra nas Figuras 2 e 3.

Figura 2 - Modelo Representativo do *Data Center* 3D



Fonte: Elaborado pelo autor.

Figura 3 - Dimensões da estrutura da maquete em tamanho real

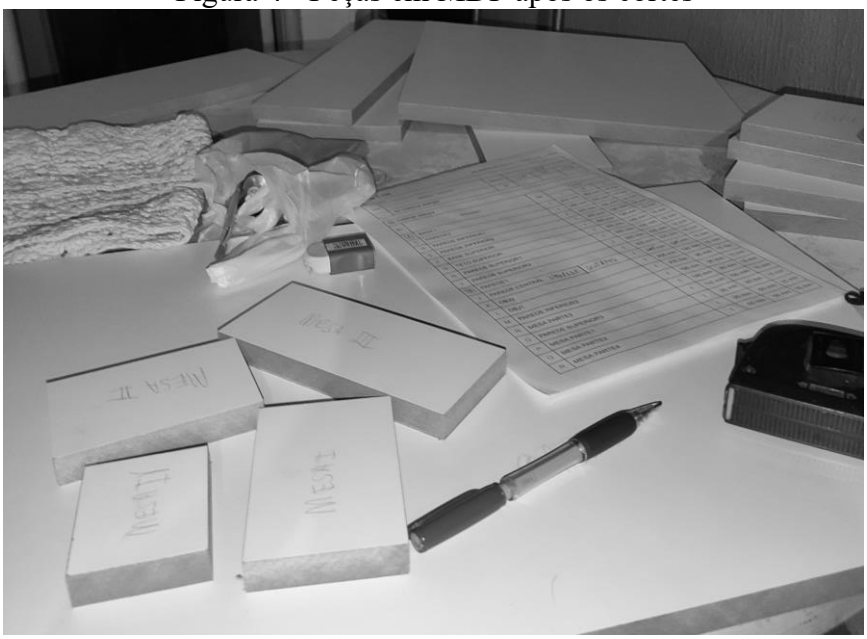


Fonte: Elaborado pelo autor.

3.2.3. Maquete

A última etapa é a construção do modelo representativo do *Data Center*, onde é feita a confecção das peças e a montagem da maquete. Com as medidas obtidas no modelo 3D, foram feitos cortes em uma chapa de madeira MDF (*Medium Density Fiberboard* - Fibra de Média Densidade) e foram geradas as peças da maquete. Em foi feita a montagem das peças seguindo o modelo 3D proposto.

Figura 4 - Peças em MDF após os cortes



Fonte: Elaborado pelo autor.

A Figura 4, mostra as peças de MDF antes da sua montagem. A montagem foi feita em seguida, com o auxílio de uma furadeira, parafusos e cola de madeira, resultando na estrutura da Figura 5.

Figura 5 - Maquete logo após a montagem

Fonte: Elaborado pelo autor.

Figura 6 - Maquete finalizada

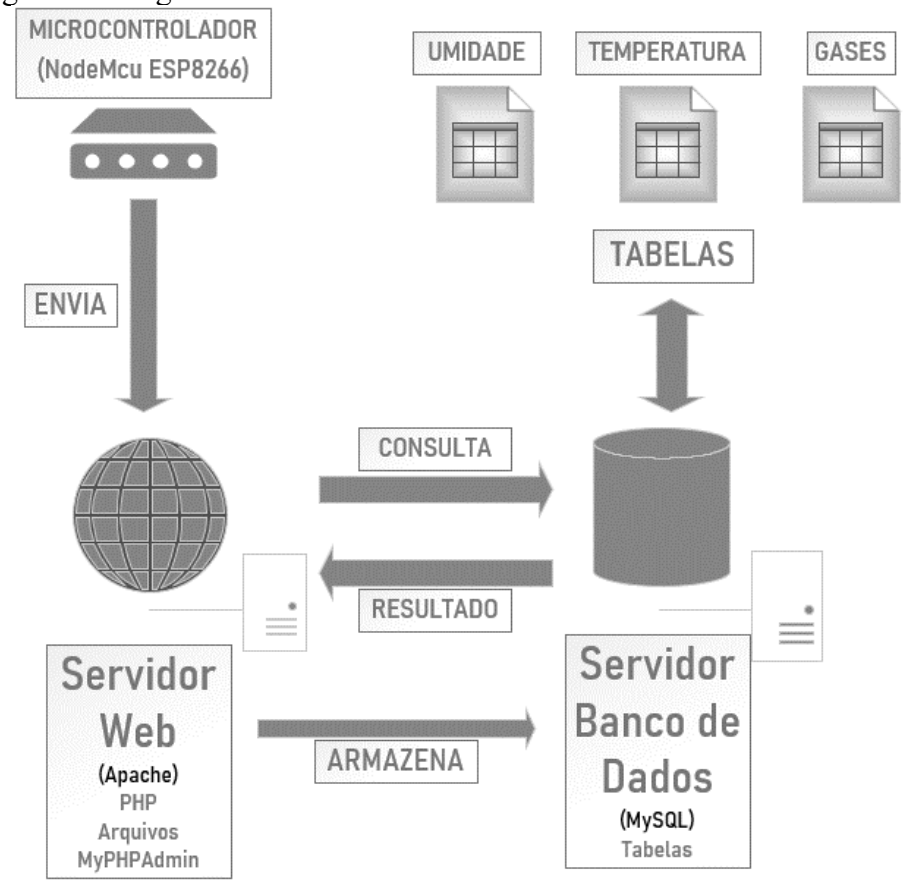
Fonte: Elaborado pelo autor.

Na Figura 6, a maquete pode ser visualizada em seu estado final. Após ter sido feito um acabamento nas peças para melhorar a identificação visual dos elementos do modelo representativo do *Data Center*, citados na seção 3.2.1 desta monografia. Além disso, foram colocadas etiquetas com os nomes de cada elemento.

3.3. Construção do Projeto do Microcontrolador

Esta seção detalha todas as etapas de construção do projeto do microcontrolador, que é o *hardware* responsável por realizar o monitoramento do ambiente de testes simulado pela maquete. O microcontrolador gerencia a captura dos dados fornecidos pelos sensores e repassa para o servidor de banco de dados, porém de forma indireta, através de outro servidor *web*, que por fim serão exibidos pela plataforma *web* de monitoramento. No diagrama da Figura 7, é observado o fluxo dos dados obtidos pelo microcontrolador, que será mais bem detalhado nas próximas seções.

Figura 7 - Diagrama do fluxo de dados entre microcontrolador e servidores



Fonte: Elaborado pelo autor.

3.3.1. Componentes

Para a construção do dispositivo de monitoramento, foram utilizados os seguintes componentes eletrônicos, avaliados em R$ 107,70 no dia da compra:

1. Microcontrolador Wi-Fi ESP8266 NodeMcu ESP-12E (R$ 37,90);
2. Sensor de Gás MQ-2 Inflamável e Fumaça (R$ 18,90);
3. Módulo Sensor de Umidade e Temperatura DHT11 (R$ 15,90);
4. Cabeamento, Protoboard e demais componentes (R$ 35,00).

A relação custo benefício deste projeto de monitoramento é propositalmente muito boa e viável para qualquer tipo de estudo. O custo dos componentes é considerado baixo para as vantagens que eles nos proporcionam. O microcontrolador, por ser capaz de se comunicar com dispositivos periféricos, com sensores e ter uma conexão Wi-Fi (*Wireless Fidelity* - Fidelidade sem fio), ele pode ser utilizado para diversas outras funções não relacionadas com esse trabalho em si, garantindo assim ainda a reutilização dos componentes para outros trabalhos futuros.

Os sensores utilizados neste projeto são apenas para propósitos experimentais. Não é recomendado o uso desses sensores em projetos reais e que envolva ambientes reais em grande escala como os próprios *Data Centers*, podendo causar problemas de segurança devido à sua má adequação ao ambiente proposto.

3.3.1.1. Microcontrolador Wi-Fi ESP8266 NodeMcu ESP-12E

Um microcontrolador é um pequeno dispositivo inteligente, constituído de uma CPU, memória de dados e periféricos (portas de Entrada e Saída), sendo capaz de desempenhar diversas funções de controle. Pode-se controlar um sistema que faz uso de medições de sensores, um sistema de acionamento de LEDs (*Light Emitting Diode* - Diodo Emissor de Luz), movimentos de robôs, controlar dispositivos periféricos de vários tipos como teclados e telas e entre outros (MIYADAIRA, 2009).

A princípio os microcontroladores não possuem nenhuma função de fábrica, a ideia é que ele seja programado para realizar suas funções de acordo com a necessidade do programador. Além disso, para fins didáticos, eles geralmente são adquiridos montados em pequenas placas de circuito integrado que já possui diversos recursos para entrada e saída de dados e periféricos que auxiliam na programação e na alimentação do dispositivo, como é o caso da placa NodeMcu ESP-12E utilizada neste trabalho (MIYADAIRA, 2009).

O módulo Wi-Fi ESP8266 NodeMcu é uma mini placa micro controladora, composta pelo chip ESP8266, que permite a comunicação de dados via Wi-Fi através de uma antena embutida. O microcontrolador conta também com um conector micro-USB para conexões diretas com o computador para a gravação do código, além de 11 pinos de entrada e saída analógico e digital, e um módulo ESP-12E que possui um aumento considerável no alcance do sinal wireless (OLIVEIRA, 2017).

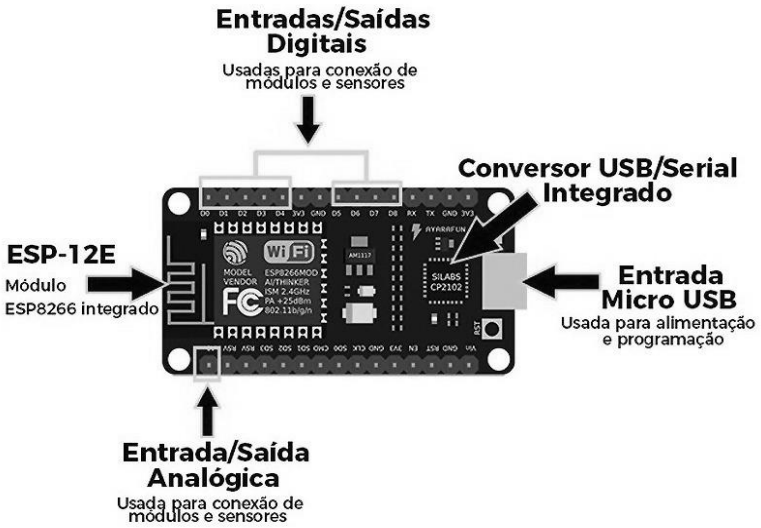
Especificações:

- ESP8266 ESP-12E;
- Wireless padrão 802.11 b/g/n;
- Antena embutida;
- Conector micro-USB;
- Modos de operação: STA/AP/STA+AP;
- Suporta 5 conexões TCP/IP;

- Portas GPIO: 11;
- GPIO (*General Purpose Input/Output* - Entrada e Saída de Uso Geral) com funções de PWM, I2C, SPI e etc.;
- Tensão de operação: 4,5 ~ 9V
- Taxa de transferência: 110-460800 bps;
- Suporta Upgrade remoto de *firmware;*
- Conversor analógico digital (ADC);
- Distância entre pinos: 2,54 mm;
- Dimensões: 49 x 25,5 x 7 mm.

Na Figura 8, pode-se observar a disposição das portas da placa NodeMcu ESP8266 ESP-12E e seus componentes.

Figura 8 - Módulo Wi-Fi ESP8266 NodeMcu ESP-12E



Fonte: OLIVEIRA, 2017.

3.3.1.2. Sensor de Gás MQ-2

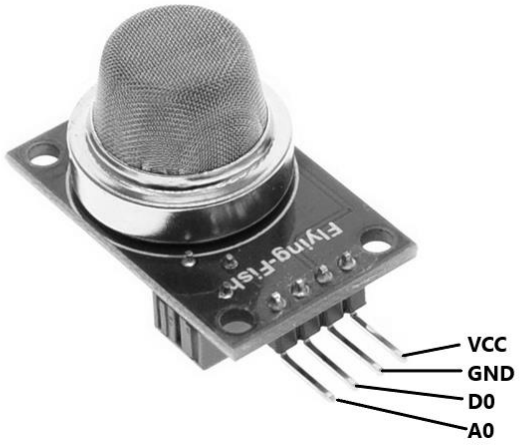
O sensor de gás MQ-2 é um módulo eletrônico desenvolvido com a finalidade de detectar certas concentrações de gases combustíveis e fumaça no ar. A partir da detecção pelo sensor, o microcontrolador ligado ao módulo será notificado e poderá tomar as ações necessárias para fornecer as medições ao usuário (OLIVEIRA, 2018).

Especificações:

- Modelo: MQ-2;
- Detecção de gases inflamáveis: GLP (Gás Liquefeito de Petróleo), Metano, Propano, Butano, Hidrogênio, Álcool, Gás Natural e outros inflamáveis;
- Detecção de fumaça;
- Concentração de detecção: 300-10.000ppm;
- Tensão de operação: 5V;
- Sensibilidade ajustável via potenciômetro;
- Saída digital e analógica;
- Fácil instalação;
- Comparador LM393;
- LED indicador para tensão;
- LED indicador para saída digital;
- Dimensões: 32 x 20 x 15mm.

Na Figura 9, pode-se visualizar a pinagem do sensor. Ele possui uma alimentação VCC (Voltagem em Corrente Contínua) de 5V, o pino neutro GND (*Graduated Neutral Density* - Filtro Graduado de Densidade Neutra), uma saída digital D0 e uma saída analógica A0.

Figura 9 - Sensor de gás inflamável MQ-2



Fonte: OLIVEIRA, 2018 (imagem original modificada).

3.3.1.3. Sensor de Temperatura e Umidade DHT11

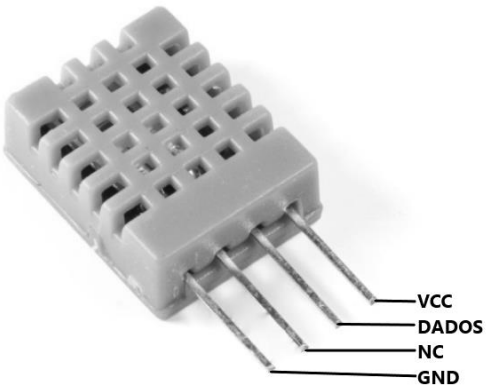
O sensor DHT11 é um sensor de temperatura e umidade que permite fazer medições de umidade entre 20% a 90% e temperaturas entre 0º C a 50º C. Muito usado para projetos

experimentais, ele conta com um termistor do tipo NTC (*Negative Temperature Coefficient* - Coeficiente de Temperatura Negativo) e um sensor de umidade HR202, ambos conectados a um controlador de 8-bits. Seu circuito interno faz a leitura do ambiente com os sensores e envia o sinal ao microcontrolador através de uma via serial de mão única (pinagem de DADOS). Ele possui 4 pinos ao todo, o pino de alimentação VCC de 3V a 5V, o neutro GND, o pino de dados, e um pino vazio NC (*Not Conected* - Não Conectado). Sua pinagem pode ser observada na Figura 10 (THOMSEN, 2013).

Especificações:

- Modelo: DHT11;
- Faixa de medição de umidade: 20 a 90% UR;
- Faixa de medição de temperatura: 0º a 50ºC;
- Alimentação: 3-5VDC (5,5VDC máximo);
- Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA;
- Precisão de umidade de medição: ± 5,0% UR;
- Precisão de medição de temperatura: ± 2,0º C;
- Tempo de resposta: 2s;
- Dimensões: 23 x 12 x 5mm (incluindo terminais);

Figura 10 - Sensor DHT11

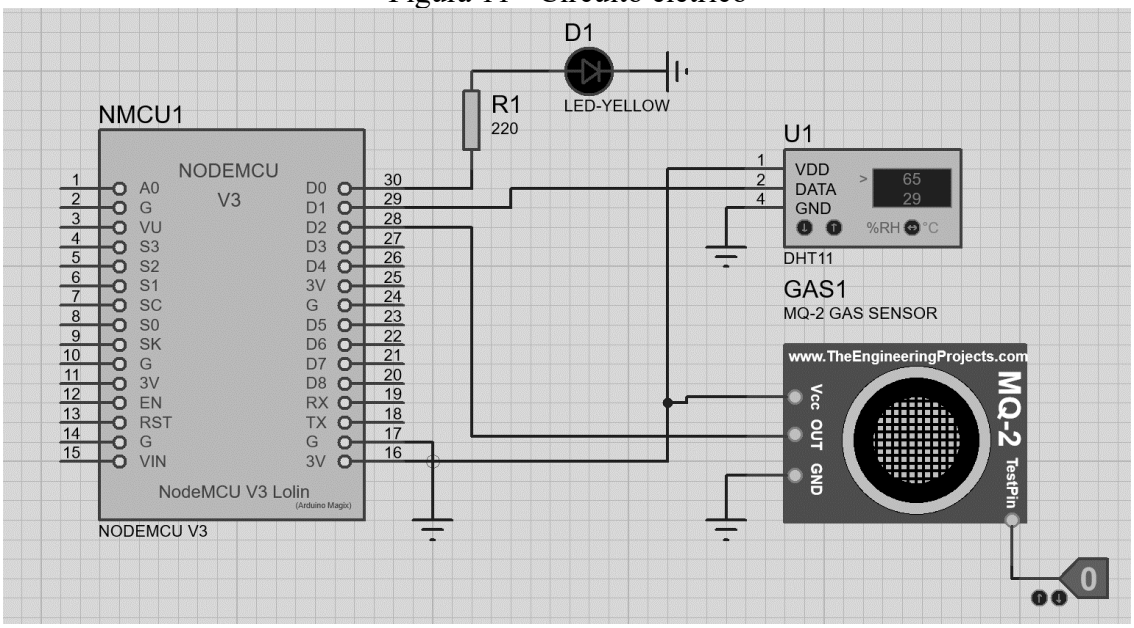


Fonte: THOMSEN, 2013 (imagem original modificada).

3.3.2. Circuito Elétrico

O circuito elétrico trabalha a ligação dos componentes elétricos citados na seção 3.3.1. desta monografia. A ligação é feita através de fios condutores ligados no microcontrolador e nos sensores, como mostra a Figura 11.

Figura 11 - Circuito elétrico



Fonte: Elaborado pelo autor.

O microcontrolador NodeMcu ESP8266 possui 11 pinos digitais de entrada e saída de dados, porém neste projeto foram utilizados apenas três pinos para a comunicação com os dispositivos periféricos.

Como mostra a Figura 11, o primeiro pino, observando de baixo para cima, trata se da alimentação provinda do microcontrolador (3,3 V) para o resto do circuito. Esse pino é responsável por fornecer a energia necessária para todos dispositivos periféricos. O pino “3V” faz a alimentação dos dois sensores utilizados neste trabalho, o sensor DHT11 (sensor de temperatura e umidade) e o MQ-2 (sensor de gás).

O segundo pino é o “G” ou GND, é a pinagem negativa (0 V) do circuito, que faz referência a alimentação do circuito. Este pino também deve estar ligado em todos os dispositivos para a circulação da corrente elétrica.

Os pinos “D2” e “D1” são os responsáveis pela troca de informações entre os sensores e o microcontrolador. O sensor de gás envia informações sobre a detecção de gases e fumaça através da porta “OUT” (do inglês, “saída”) ligada no pino “D2” (entrada), responsável por receber os dados. Já o sensor de temperatura e umidade envia as informações do ambiente através da porta “DATA” (do inglês, “dados”, onde também é a saída dos dados) ligada no pino “D1” (entrada), também responsável por receber os dados. Por fim, o pino “D0” representa apenas um sinal de saída luminoso para a evidenciar a detecção de gases no sistema, e está ligado apenas a uma resistência de 220 ohms e um LED.

Logo, conclui-se que os pinos “D2” e “D1” atuam como pinos de entrada de dados, dados estes recebidos pelos sensores, já o pino “D0” atua como saída de dados, pois ele vai gerar o sinal luminoso dependendo da resposta obtida pelo sensor de gás.

3.3.3. Programação

A programação do microcontrolador é feita através de uma conexão via micro-USB no dispositivo, o programa é carregado para dentro da placa e armazenado em sua memória interna, sendo uma vez carregado o dispositivo pode ser reiniciado e desligado sem que perca sua programação. No entanto, se for de necessidade de o programador atualizar o programa já existente ou limpar a memória do dispositivo, basta inserir um novo código sobrepondo o anterior ou realizar o *reset* de sua memória (MONK, 2017).

O microcontrolador NodeMcu ESP8266 ESP-12E, é um *hardware* baseado no padrão de placas do Arduino e sua programação pode ser feita na mesma IDE. Os Arduinos são uma plataforma de prototipagem com mini placas microcontroladoras de projeto aberto (*open source*), ou seja, qualquer um pode modificar, melhorar e personalizá-los. Já o NodeMcu ESP8266 ESP-12E é uma variação melhorada do padrão surgido a partir do Arduino, apresentando suas semelhanças e diferenças (THOMSEN, 2016).

A programação do microcontrolador neste trabalho está estruturada da seguinte forma, como pode ser visto no Exemplo da Figura 12:

1. Inclusão das bibliotecas;
2. Mapeamento do *Hardware*;
3. Declaração de variáveis e objetos;
4. Configuração inicial;
5. Estrutura de repetição.

Figura 12 - Exemplo de estrutura utilizada na programação

```
1  // ----------------- BIBLIOTECAS ---------------
2  #include <Arduino.h>
3
4  // -------- MAPEAMENTO DE HARDWARE -------------
5
6  #define LED_INT 2
7  #define MQ2_PIN 4
8
9  // ----------------- OBJETOS ------------------
10 DHT_Unified dht(DHTPIN, DHTTYPE);
11
12 // ----------C------ VARIAVEIS -----------------
13 uint32_t delayMS = 500;
14 float temperatura = 0, umidade = 0;
15 String gas;
16
17 void setup() {
18
19   // CÓDIGO DE CONFIGURAÇÃO INICIAL
20
21 }
22
23 void loop() {
24
25   // ESTRUTURA DE REPETIÇÃO
26 }
```

Fonte: Elaborado pelo autor.

Nas próximas seções será trabalhado de forma mais detalhada cada parte da construção do código, desde a criação das estruturas até a parte de simulação do microcontrolador em um ambiente virtual.

3.3.3.1. Inclusão das Bibliotecas

As bibliotecas são um conjunto de códigos previamente preparados com funções, parâmetros e variáveis salvas em arquivos separados na própria IDE (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado) do Arduino, geralmente preparados pela comunidade ou pela própria IDE, que podem ser utilizados pelo programador sem que tenham necessidade de aparecer no programa principal. São muito úteis pois não poluem o código principal e nos permite otimizar tempo na hora de preparar o código, reutilizando algo já pronto (VIDAL, 2017).

Neste trabalho é utilizado uma biblioteca para comandos básicos do próprio Arduino, além de outras três para a comunicação com os sensores e mais duas para a conexão e comunicação via Wi-Fi.

Figura 13 - Bibliotecas

```
1  // ----------------- BIBLIOTECAS ---------------
2  // Propria do Arduino
3  #include <Arduino.h>
4
5  // Uso de sensores
6  #include <Adafruit_Sensor.h>
7  #include <DHT.h>
8  #include <DHT_U.h>
9
10 // Comunicação WIFI
11 #include <ESP8266HTTPClient.h>
12 #include <ESP8266WiFi.h>
13
```

Fonte: Elaborado pelo autor.

Para a inclusão das bibliotecas, deve-se inserir antes “#include” e na frente o seu nome, como pode ser visto na Figura 13, lembrando que o arquivo da biblioteca deve existir em algum lugar dentro da pasta de bibliotecas da IDE.

3.3.3.2. Mapeamento do Hardware

O mapeamento do *hardware* consiste em definir previamente a identificação ou o nome de cada pinagem que será utilizada pelo programa. Isso facilita na hora de identificar a função de cada pino e o que ele vai representar.

Figura 14 - Mapeamento do *Hardware*

```
14 // --------------- MAPEAMENTO DE HARDWARE ---------------
15 #define DHTPIN 5     // Pino Digital conectado ao sensor DHT
16
17 // Definindo o tipo do sensor:
18 #define DHTTYPE    DHT11     // DHT 11
19 #define LED_PIN 16   // Pino do LED
20 #define LED_INT 2
21 #define MQ2_PIN 4
22
```

Fonte: Elaborado pelo autor.

Como mostra na Figura 14, o mapeamento é feito inserindo “#define”, o nome que identifica a pinagem e a numeração do pino. Com isso, ao invés de se referir ao número do pino no código, basta chamá-lo pelo nome definido. Por exemplo, na linha 20, o pino 2 representa um LED interno do próprio dispositivo, logo, com o mapeamento basta chamá-lo de “LED_INT”, para facilitar sua identificação.

3.3.3.3. Declaração de Variáveis e Objetos

As variáveis são um recurso da programação utilizadas para armazenar dados dentro da memória interna do dispositivo. Esses dados podem ser usados e modificados dentro do programa, bastando definir antes o seu tipo de dado, que é o tipo de informação que será armazenada dentro dela, e o seu identificador (seu nome) (MONK, 2017).

Os objetos, assim como as variáveis são dados que podem ser manipulados, porém os objetos devem seguir um “molde” ou uma classe à qual eles pertencem, compartilhando características e funções próprias. Assim, conclui-se que os objetos surgem a partir de uma instância de uma classe (MONK, 2017).

Figura 15 - Declaração de variáveis e objetos

```
23 // --------------------- OBJETOS -----------------------
24 DHT_Unified dht(DHTPIN, DHTTYPE);
25
26 // --------------- VARIAVEIS ---------------
27 uint32_t delayMS = 500;
28 float temperatura = 0, umidade = 0;
29 String gas;
30 // ACESSO AO WIFI
31 const char* ssid = "XXXXXX";
32 const char* password = "XXXXX";
33
```

Fonte: Elaborado pelo autor.

Como observado no código da Figura 15, foi utilizado neste trabalho uma variável para cada dado provindo dos sensores, temperatura umidade e gases, variáveis para o acesso do Wi-Fi e outra para representação de *delay* (tempo de atraso). Além disso, também foi criado um objeto para representar uma instância da classe da biblioteca do sensor de temperatura e umidade o DHT11.

3.3.3.4. Configuração Inicial

Nessa etapa do código, é onde são feitas as configurações iniciais do programa, como a configuração da taxa de transferência em *bits* para a comunicação serial, declaração dos valores de variáveis, definição das portas como entrada ou saída, configurações de Wi-Fi e outras (MONK, 2015).

Figura 16 - Configuração inicial

```
34 void setup() {
35
36   Serial.begin(115200); //Conexão Serial
37
38   // CONFIGURAÇÃO DO WIFI
39   pinMode(LED_INT, OUTPUT);
40   digitalWrite(LED_INT, 0);
41   WiFi.begin(ssid, password);//Inicia a conexão
42   // Espera a conexão do WIFI completar
43   while (WiFi.status() != WL_CONNECTED) {
44     delay(500);
45     Serial.print(".");
46   }
47   Serial.println("");
48   Serial.print("Connected to ");
49   Serial.println(ssid);
50   Serial.print("IP address: ");
51   Serial.println(WiFi.localIP());
52   digitalWrite(LED_INT, 1);
53   delay(1000);
54   digitalWrite(LED_INT, 0);
55   delay(1000);
56   digitalWrite(LED_INT, 1);
57   delay(1000);
58   digitalWrite(LED_INT, 0);
59   delay(1000);
60   digitalWrite(LED_INT, 1);
61   // CONFIGURAÇÃO DOS SENSORES
62   dht.begin();
63   pinMode(LED_PIN, OUTPUT);
64   pinMode(MQ2_PIN, INPUT); //DEFINE O PINO COMO ENTRADA
65   digitalWrite(LED_PIN, LOW); //LED INICIA DESLIGADO
66 }
```

Fonte: Elaborado pelo autor.

Como mostra o código da Figura 16, linha 34, essa etapa é definida na função “setup()”, onde todas as configurações são inseridas dentro de suas chaves. Essa função é executada apenas uma vez no início do programa.

Analisando as linhas do código acima, pode-se destacar o seguinte:

- Linha 36: define uma taxa de transmissão serial de 115200 bps, valor este recomendado pela própria fabricante do microcontrolador;
- Linhas 38 a 60: define as configurações para a conexão com o Wi-Fi, além de saídas no console e sinais luminosos no LED interno do microcontrolador para visualizar se a conexão foi estabelecida com sucesso ou não;
- Linhas 61 a 65: configuração inicial dos sensores e suas portas. No caso, inicializando o objeto dht, necessário para a utilização do sensor de temperatura e umidade. É definido também as portas de comunicação com os sensores como sendo entrada ou saída de dados.

3.3.3.5. Estrutura de Repetição

Na estrutura de repetição se encontra todo o código principal, aqui está presente todas as ações que o microcontrolador irá fazer. Trata-se de um de um laço de repetição contínuo que será executado até que algum comando de "parar" seja executado ou o dispositivo desligue (ARDUINO, 2020).

Essa estrutura é identificada como "loop()" no código, assim como mostra na Figura 17, diferente da função "setup()" que irá executar apenas uma vez, essa função como, já foi dito, irá ficar em execução sem parar. Por isso é muito comum utilizar funções do tipo "delay()" no fim do *loop*, ela é responsável por pausar as atividades presentes dentro da estrutura de repetição, sendo utilizada para facilitar a visualização dos dados obtidos e para que os sensores tenham tempo de realizar as medições (ARDUINO, 2020).

Figura 17 - Estrutura de repetição, parte 1

```
68 void loop() {
69
70   digitalWrite(LED_INT, 1);
71
72   // MEDIÇÃO DE GASES E FUMAÇA
73   if (digitalRead(MQ2_PIN) == LOW) { //SE A LEITURA DO PINO FOR IGUAL A LOW, FAZ
74     digitalWrite(LED_PIN, HIGH); //ACENDE O LED
75     gas = "S";
76   } else { //SENÃO, FAZ
77     digitalWrite(LED_PIN, LOW); //APAGA O LED
78     gas = "N";
79   }
80
```

Fonte: Elaborado pelo autor.

Figura 18 - Estrutura de repetição, parte 2

```
80
81
82  // LEITURA DO SENSOR DE TEMPERATURA
83  sensors_event_t event;
84  dht.temperature().getEvent(&event);
85  // Verificando se o retorno é valido
86  if (isnan(event.temperature)) {
87    Serial.println("Erro ao ler a temperatura!");
88  }
89  else {
90    Serial.print("Temperatura: ");
91    Serial.print(event.temperature);
92    Serial.println("°C");
93    temperatura = event.temperature;
94  }
95
96  // LEITURA DO SENSOR DE UMIDADE
97  dht.humidity().getEvent(&event);
98  // Verificando se o retorno é valido
99  if (isnan(event.relative_humidity)) {
100   Serial.println("Erro ao ler a humidade!");
101 }
102 else {
103   Serial.print("Humidade: ");
104   Serial.print(event.relative_humidity);
105   Serial.println("%");
106   umidade = event.relative_humidity;
107 }
108
```

Fonte: Elaborado pelo autor.

Figura 19 - Estrutura de repetição, parte 3

```
109
110 // ENVIANDO OS DADOS VIA REQUISIÇÃO GET (PHP) AO BANCO DE DADOS
111 if (WiFi.status() == WL_CONNECTED) { //checando o status da conexão WiFi
112   HTTPClient http;    //Declarando o objeto da classe HTTPClient
113   //Destino da requisição
114   http.begin("http://192.168.0.7/get/get.php?temperatura="
115   + String(temperatura)
116   + "&umidade=" + String(umidade) + "&gas=" + gas);
117
118   int httpCode = http.GET();   //Envia a requisição
119   String payload = http.getString();     //Resposta payload
120
121   // Status de resposta. Exemplos: 200 --> OK, 404,-1 --> ERRO
122   Serial.println("Status da conexão = " + httpCode);
123
124   http.end();  //Fechando a conexão
125 } else {
126   Serial.println("Error in WiFi connection");
127 }
128
129 // Exibição no console
130 Serial.println("Temperatura = " + String(temperatura));
131 Serial.println("Umidade = " + String(umidade));
132 Serial.println("Gas ou fumaca = " + gas);
133
134 delay(5000);  //Enviando a requisição a cada 5 segundos
135
136 }
```

Fonte: Elaborado pelo autor.

Analisando as linhas do código das Figuras 17, 18 e 19, pode-se destacar o seguinte:

- Linhas 72 a 79 (Figura 17): código para a leitura do sensor de gases e fumaça, caso o sensor identifique algum gás ele envia uma resposta para microcontrolador que caso seja positiva a variável “gas” recebe o texto “S”, e caso contrário recebe “N”;
- Linhas 82 a 107 (Figura 18): leitura de temperatura e umidade. Como utilizam o mesmo sensor (DHT11) a leitura é feita de modo semelhante, utilizando o mesmo objeto criado a partir de uma classe da biblioteca do sensor. Com as funções “event.temperature” e “event.humidity” pode-se capturar os dados do sensor e armazená-los diretamente nas variáveis de temperatura e umidade já no formato de graus celsius e porcentagem de umidade relativa do ar;
- Linhas 110 a 127 (Figura 19): código responsável pelo envio de dados para o banco de dados. Essa parte será mais bem trabalhada na seção 3.4, mas é enviado os dados recolhidos pelos sensores para um servidor *web* hospedado em uma máquina na mesma rede, enviando uma requisição GET, que é um dos métodos de comunicação presente no protocolo HTTP;
- Linha 134 (Figura 19): delay de 5000 milissegundos ou 5 segundos. Representa a pausa ou o intervalo para repetir o laço “loop()”, para obter e enviar os novos dados.

Agora, com o programa pronto, basta colocá-lo em prática, realizando simulações e os testes práticos. Para isso basta carregar o programa na placa microcontroladora via *software* ou via conexão USB.

3.3.4. Simulação

Toda e qualquer simulação é essencial para o desenvolvimento de sistemas embarcados (sistemas independentes de propósito específico) e circuitos elétricos em geral, atuando como uma ferramenta de verificação das ligações elétricas e o seu funcionamento nos mais diversos projetos. Antes de realizar os testes práticos, é importante visualizar o comportamento dos dispositivos e suas ligações no ambiente virtual, sendo capaz de se preparar melhor para o que está por vir, evitando possíveis problemas técnicos que possam atingir e danificar os dispositivos.

Existem diversas ferramentas de simulação, entre elas está o *Proteus Design Suite*, que além de realizar as simulações, cria projetos esquemáticos dos circuitos elétricos do zero e

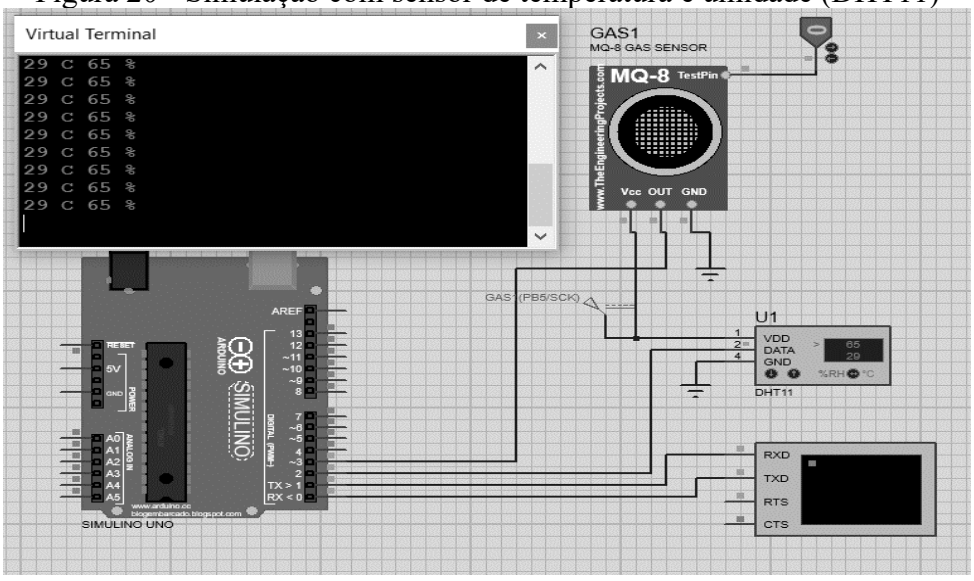
esquemas de ligações de PCB (*Print Circuit Board* - Placa de Circuito Impresso), além de fornecer recursos extras importantes, como um terminal virtual, osciloscópio, gerador de sinais, amperímetros, voltímetros, diversos microcontroladores e diversos outros dispositivos eletrônicos (MADEIRA, 2015).

Devido ao fato de que esta e diversas outras ferramentas não possuem o microcontrolador específico utilizado neste projeto, foram feitas algumas adaptações para a realização da simulação do microcontrolador e a comunicação com os sensores. Ao invés de usar o microcontrolador NodeMcu ESP8266 ESP-12E, foi utilizado outro microcontrolador semelhante, o Arduino Uno R3, que é um modelo de referência para os outros demais Arduinos (MURTA, 2018).

Conforme já foi mencionado na sessão 3.3.1.1 e na sessão 3.3.3 dessa monografia, o modelo NodeMcu nada mais é que uma variação melhorada do Arduino padrão. Logo, para sua programação é utilizada a mesma IDE e a mesma linguagem, implicando em um código semelhante e com resultados simulados idênticos (MURTA, 2018).

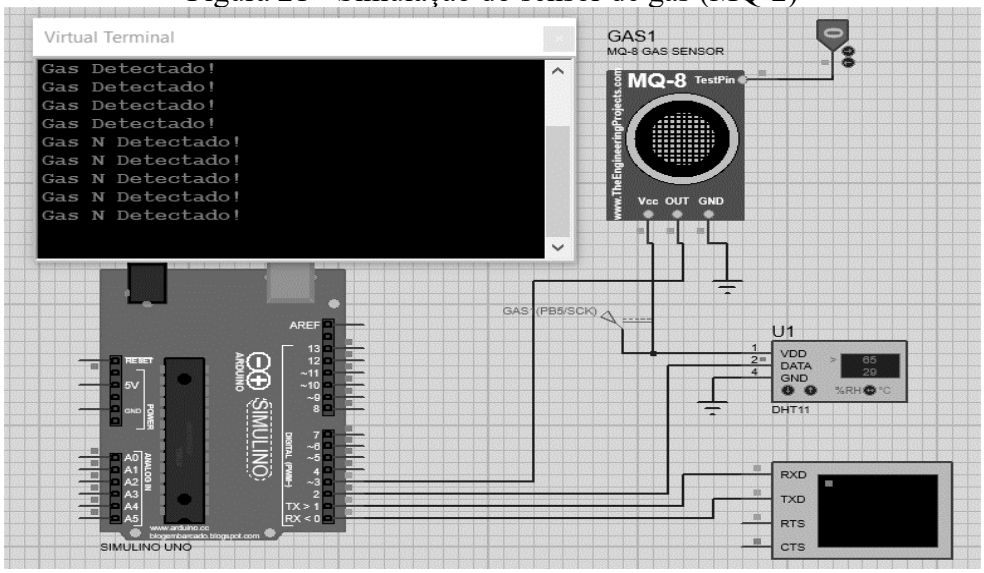
Foram feitas duas simulações na ferramenta *Proteus Design Suite* (versão 8.9), a primeira foi a simulação do sensor de temperatura e umidade DHT11, e a segunda foi com o sensor de gás MQ-2, como é mostrado nas Figuras 20 e 21, respectivamente.

Figura 20 - Simulação com sensor de temperatura e umidade (DHT11)



Fonte: Elaborado pelo autor.

Figura 21 - Simulação do sensor de gás (MQ-2)



Fonte: Elaborado pelo autor.

A simulação é feita com o auxílio de um terminal virtual disponibilizado pela ferramenta, onde são exibidos os resultados capturados pelo próprio microcontrolador através das ligações feitas nas portas TX e RX, que refere-se respectivamente as pinagens de recepção e transmissão de dados do microcontrolador Arduino Uno R3, que recebe os valores obtidos dos sensores e transmite ao terminal.

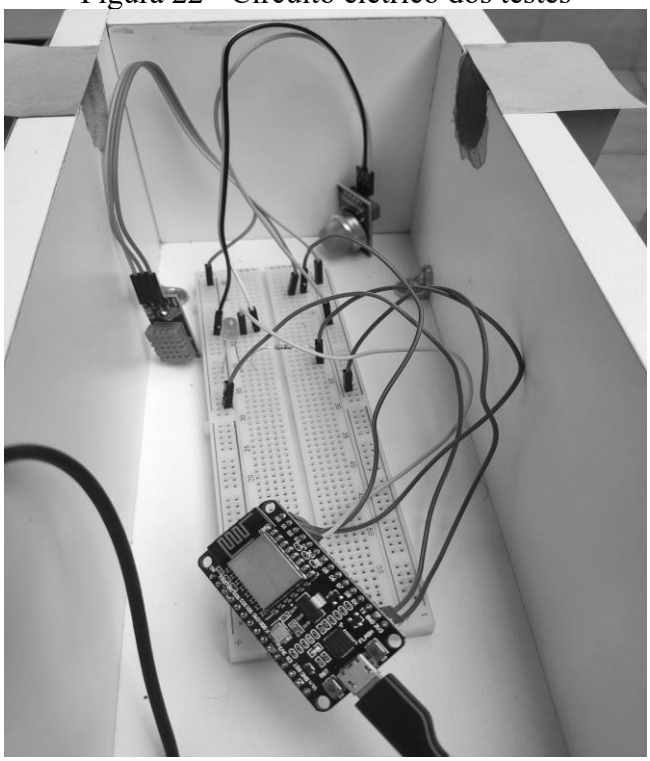
O código e a biblioteca utilizada para essa simulação são os mesmos demonstrados anteriormente na sessão 3.3.3 desta monografia. O programa foi carregado para o microcontrolador via *software*, através de um arquivo binário já compilado.

Por haver limitações no ambiente de teste virtual, parte da simulação teve que ser feita no ambiente real, já com o circuito montado ou parte dele. Os testes práticos envolveram tanto a captura dos dados dos sensores, como também a parte de conexão do Wi-Fi e o banco de dados, partes essas muito limitadas dentro do ambiente virtual.

3.3.5. Circuito na Prática

As ligações dos componentes eletrônicos na prática é a mesma utilizada na Figura 11, da seção 3.3.2 desta monografia. Para fins didáticos as ligações dos componentes eletrônicos foram feitas utilizando uma *protoboard* (matriz de contatos, placa com furos e conexões internas para montagem de circuitos) e *jumpers* (fios), facilitando a realização dos testes e na melhor visualização das ligações, como mostra a Figura 22.

Figura 22 - Circuito elétrico dos testes



Fonte: Elaborado pelo auto.

Durante a simulação não é necessário fornecer alimentação para o dispositivo, porém na prática é diferente, deve existir uma fonte de alimentação de corrente contínua de 3V a 5V. Por isso foi utilizado um carregador portátil de celular que é capaz de fornecer a corrente necessária para a alimentação do microcontrolador e o circuito como um todo. Basta fornecer a energia ao microcontrolador que todos os outros componentes ligados a ele serão alimentados pela saída de 3V presente no dispositivo.

Para o funcionamento do circuito, é necessário primeiramente realizar o carregamento do programa para a memória interna do microcontrolador. Para isso é necessário conectá-lo via USB no computador ao qual a IDE do Arduino está instalada, e em seguida basta realizar a compilação e o carregamento do programa para o dispositivo. Uma vez feito, o programa já faz parte da memória do microcontrolador, agora basta que o dispositivo receba uma alimentação externa para funcionar de forma independente.

3.4. Persistência dos Dados

Agora com o dispositivo microcontrolador pronto para enviar os dados e de forma independente, é importante entender como os dados serão armazenados e como eles vão se mostrar para o cliente (usuário).

A persistência dos dados se refere ao armazenamento dos dados capturados pelo dispositivo de monitoramento, o microcontrolador. O armazenamento é essencial para uma plataforma de monitoramento, visto que através dos históricos de dados é possível gerar relatórios e análises estatísticas do ambiente monitorado.

3.4.1. Banco de Dados

A persistência dos dados é realizada através de um banco de dados (ou *database*), e se trata de um conjunto de dados relacionados, que são coletados e armazenados para uma finalidade específica (ELMASRI; NAVATHE, 2014).

Os bancos de dados podem ser gerados e mantidos manualmente, como em uma lista telefônica, ou um catálogo de filmes em uma locadora. Porém, atualmente a maioria dos bancos de dados são computadorizados, os dados são gerados e mantidos através de programas específicos para esse fim, os chamados sistemas gerenciadores de banco de dados (ELMASRI; NAVATHE, 2011).

3.4.2. Sistema Gerenciador de Banco de Dados

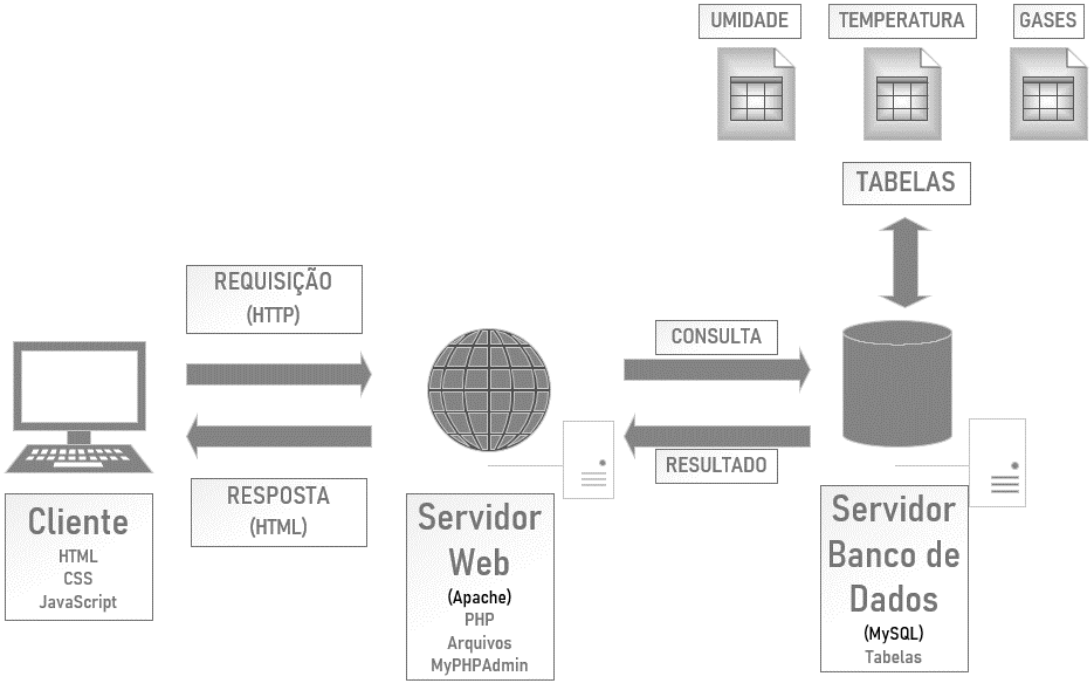
O SGBD (Sistema Gerenciador de Banco de Dados) é um *software* que permite definir, construir, manipular e compartilhar gigantescas bases de dados entre seus usuários e suas aplicações. O termo “definir”, se refere a etapa de especificar o tipo de dado e suas características, o termo “construir” é o processo de armazenamento, já o termo “manipular”, trata-se das funções de consulta e atualização dos dados, e por fim, o termo “compartilhar” refere-se ao acesso aos dados de forma simultânea entre as aplicações e o usuário. Como exemplos de SGBDs temos: MySQL, Oracle, PostgreSQL e entre outros (ELMASRI; NAVATHE, 2011).

O SGBD utilizado neste trabalho é o MySQL, *software* de código aberto baseado no modelo de bancos de dados relacional (dados organizados em tabelas), que utiliza uma linguagem específica para consulta de banco de dados, a SQL (*Structured Query Language* - Linguagem de Consulta Estruturada). A linguagem SQL é uma série de comandos estruturados utilizados para consultas, criação de tabelas, colunas e outras funções importantes que permite o gerenciamento do banco de dados (SILVERA, 2019).

Como todo e qualquer banco de dados, os arquivos precisam estar hospedados em um servidor. O MySQL segue o modelo cliente-servidor, como pode ser visto do diagrama da

Figura 23. O cliente envia uma requisição ao servidor que responde de acordo com o que foi solicitado. A solicitação no servidor MySQL pode ser feita usando comandos específicos do SQL ou pode ser feita através de ferramentas mais intuitivas e simples como é o caso do PHPMyAdmin, que também é utilizada neste projeto (SOUZA, 2020).

Figura 23 - Diagrama cliente-servidor



Fonte: Elaborado pelo autor.

O PHPMyAdmin é uma ferramenta de uso livre e código aberto que pode ser acessada com um servidor *web*, diretamente do navegador. Apresenta uma *interface* simples e intuitiva e se abstém do uso de código para realizar o gerenciamento do banco de dados. Ele é baseado na linguagem PHP (*Hypertext Preprocessor* - Pré-processador de Hipertexto), que já possui nativamente suporte para comunicação e conexão com o MySQL (PHPMYADMIN, 2020).

A linguagem PHP, é muito utilizada no meio *web* devido a sua variedade de recursos focados principalmente no lado dos servidores, ou *server-side*. O PHP permite criar páginas em HTML (*Hypertext Markup Language* - Linguagem de Marcação de Hipertexto) dinâmicas que se comunicam diretamente com o servidor, visto que ele é interpretado no mesmo local onde o servidor está instalado (PHP,2020).

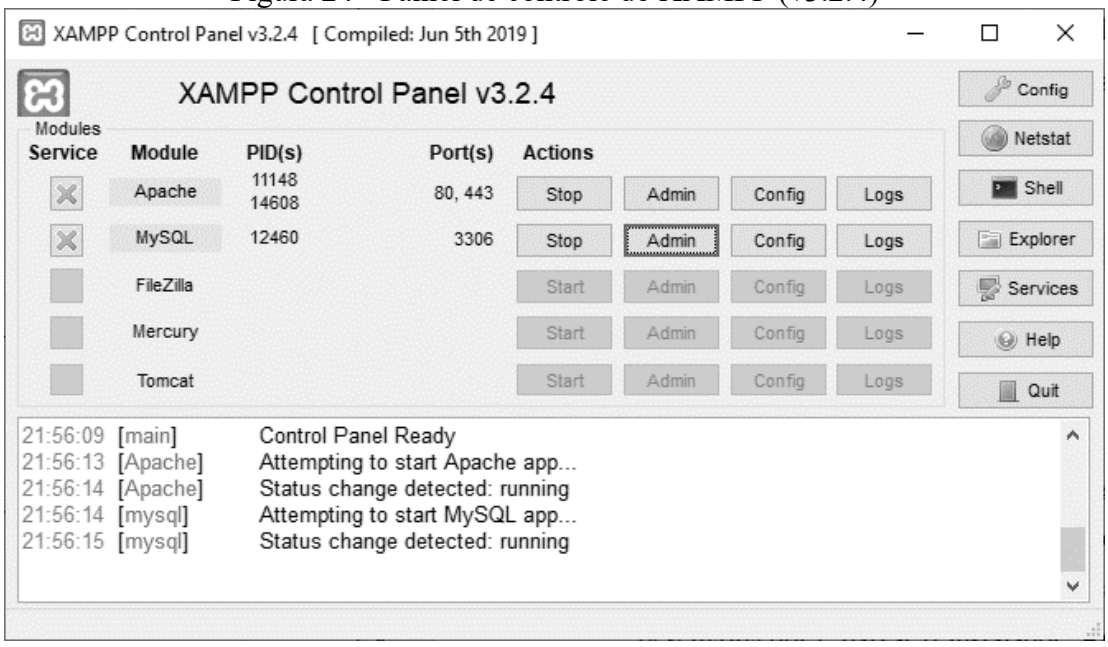
3.4.3. Criação do Servidor de Banco de Dados

A criação de um servidor de banco de dados pode ser feita de diversas formas, com *hardware*s próprios para função de servidor ou máquinas locais que podem ser transformadas em servidores. Existem diversos *softwares* que transformam a máquina em um servidor que é capaz de responder requisições de outras máquinas clientes. Assim, para facilitar o desenvolvimento do projeto, este trabalho faz o uso de um *software* chamado XAMPP, que já apresenta um pacote com os principais servidores de código abertos e gratuitos existentes, e isso inclui, o MySQL já com o PHPMyAdmin, e o servidor *web* Apache, para trabalhar melhor nas próximas seções desta monografia (XAMPP, 2020) .

O XAMPP é um *software* gratuito, para sua instalação basta acessar o *site* do desenvolvedor e baixar o instalador. Após a instalação, é apresentado um painel de controle (Figura 24) com todos os servidores disponíveis ao usuário, que permite tomar ações como:

- *Start*: inicializa o servidor correspondente;
- *Admin*: acessa a parte de gerenciamento do servidor. No caso do MySQL, ele acessa diretamente a ferramenta de administração do banco de dados, o PHPMyAdmin;
- *Config*: configuração interna do servidor. Exemplos: configurações de portas, segurança e de acesso.
- *Logs*: listas ou relatórios de erros e acessos;

Figura 24 - Painel de controle do XAMPP (v3.2.4)

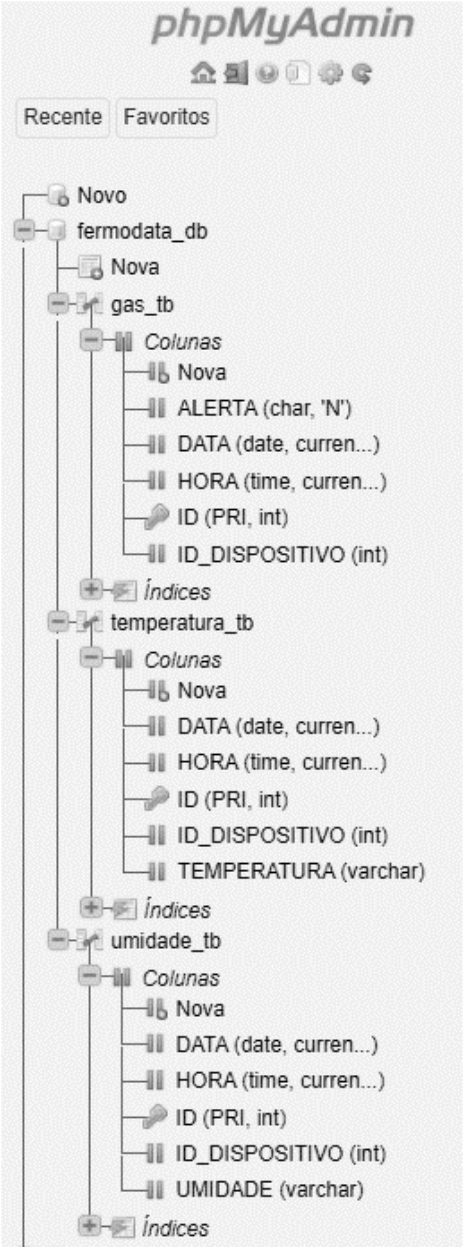


Fonte: Elaborado pelo autor.

3.4.3.1. Estrutura do Banco de Dados

Em bancos de dados como o MySQL, que segue um modelo relacional, os dados são organizados e armazenados em tabelas, e em cada tabela está um conjunto de dados específicos, onde as colunas das tabelas representam os atributos e as linhas representam os registros inseridos. As tabelas geralmente se relacionam de alguma maneira através de chaves ou índices, mas neste projeto em específico, as tabelas são utilizadas apenas para armazenar os registros de temperatura, umidade e gases (NASCIMENTO, 2015).

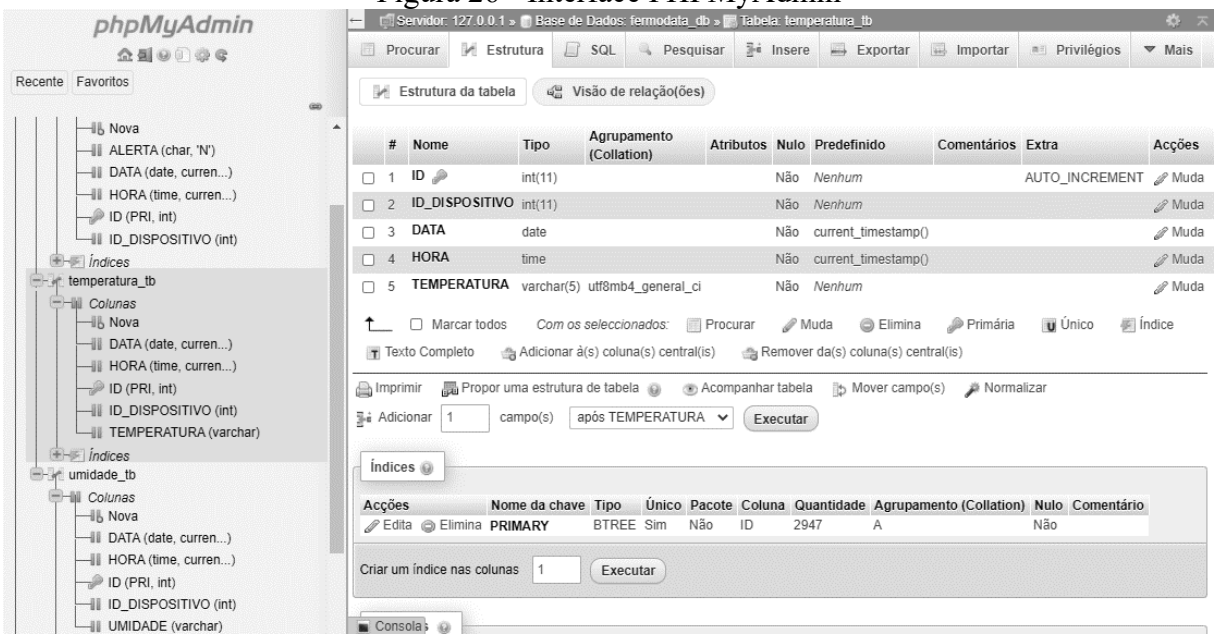
Figura 25 - Estrutura do banco de dados no PHPMyAdmin



Fonte: Elaborado pelo autor

Como mostra na Figura 25 e a Figura 26, com a *interface* da ferramenta PHPMyAdmin, é possível visualizar e gerenciar de forma mais simples e intuitiva a estrutura do banco de dados “fermodata_db” (nome que simboliza Ferramenta de Monitoramento de *Data Centers*). As tabelas estão organizadas em “gas_tb”, “temperatura_tb”, “umidade_tb”, que armazenam os dados provindos dos sensores de gases, temperatura e umidade respectivamente.

Figura 26 - Interface PHPMyAdmin



Fonte: Elaborado pelo autor.

As colunas, que representam os atributos, estão divididas em:

- ID: índice, identificador do registro;
- ID_DISPOSITIVO: identificador do dispositivo;
- DATA: data de coleta do registro;
- HORA: hora de coleta do registro;
- DADO (GÁS, TEMPERATURA, UMIDADE): dados coletados pelos sensores.

3.4.3.2. Armazenamento dos Dados

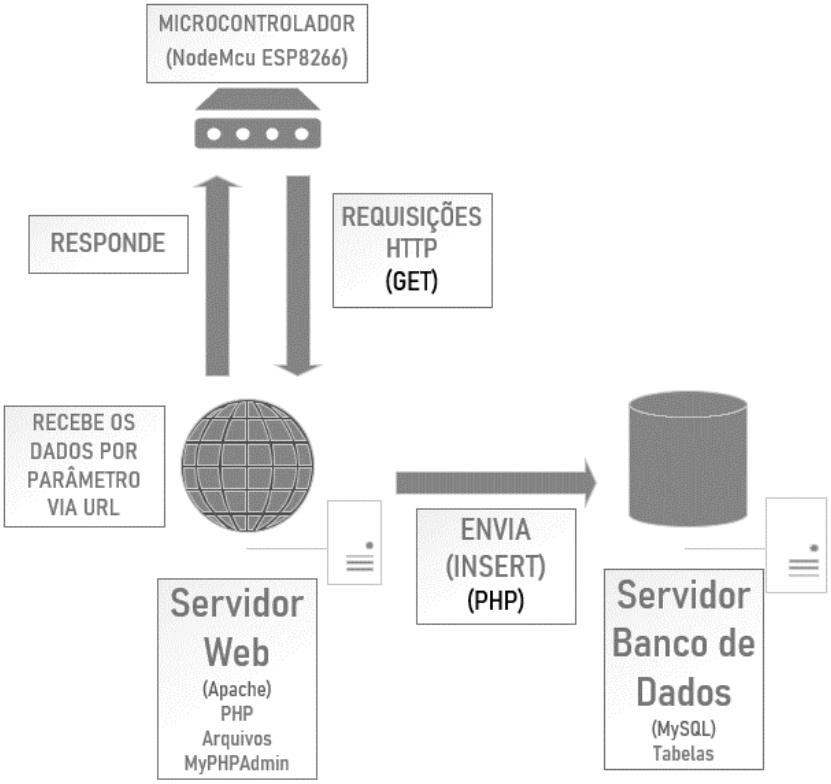
Como foi dito na seção 3.3.3.5 desta monografia, os dados são capturados pelos sensores e o microcontrolador faz o papel de alimentar os registros no banco de dados MySQL. Os dados são registrados automaticamente, a cada 5 segundos, que é o tempo de *delay* programado na função “loop()” do microcontrolador.

Antes de falar sobre a forma de envio dos dados, é importante entender que os servidores *web*, como o Apache, utilizam um protocolo de comunicação chamado HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto), que é responsável por realizar a troca de dados entre cliente-servidor através de requisições e respostas, que mostra na tela dos navegadores os dados da páginas solicitadas. O HTTP possui diversos métodos de comunicação e entre eles vale destacar o método GET, que foi o mais utilizado neste projeto (SOUZA, 2019).

O método GET consiste em uma requisição HTTP que é feita ao servidor *web*. Nela, são passados os parâmetros da requisição, como por exemplo a URL. Assim, o servidor faz a leitura dos parâmetros e responde de acordo com o que foi pedido, podendo ser uma página em HTML para o navegador interpretar (GARBIN, 2017).

Apesar de ser uma requisição, o método GET, pode ser utilizado para o envio de dados através da própria URL. Aproveitando de recursos da linguagem PHP é possível armazenar os valores presentes no parâmetro em variáveis locais, e reutilizá-las para outras funções (GUIMARÃES, 2018).

Figura 27 - Diagrama de envio de dados através de requisições GET



Fonte: Elaborado pelo autor.

Como mostra o diagrama da Figura 27, o processo de armazenamento pode resumido da seguinte forma: o microcontrolador (cliente) envia uma requisição GET ao servidor *web*, passando como parâmetro na URL (*Uniform Resource Locator* - Localizador Uniforme de Recursos) os valores de temperatura, umidade e gás, e do outro lado, o servidor *web* através de funções PHP recebe a requisição GET e armazena os valores em variáveis locais, que em seguida são e enviadas para o banco de dados MySQL, a partir de comandos em SQL do tipo "INSERT".

3.4.3.3. Lado do Cliente

O microcontrolador é capaz de enviar requisições GET ao servidor graças a sua conexão de *Internet* Wi-Fi e a uma biblioteca denominada "ESP8266HTTPClient.h", que faz do microcontrolado um dispositivo do lado cliente, capaz de enviar requisições e obter respostas do servidor, estabelecendo a comunicação cliente-servidor (ISRAEL, 2019).

Figura 28 - Código de envio da requisição GET, presente no microcontrolador

```
110  // ENVIANDO OS DADOS VIA REQUISIÇÃO GET (PHP) AO BANCO DE DADOS
111  if (WiFi.status() == WL_CONNECTED) { //checando o status da conexão WiFi
112
113    HTTPClient http;    //Declarando o objeto da classe HTTPClient
114    //Destino da requisição
115    http.begin("http://192.168.0.7/get/get.php?temperatura="
116    + String(temperatura)
117    + "&umidade=" + String(umidade) + "&gas=" + gas);
118
119    int httpCode = http.GET();   //Envia a requisição
120    String payload = http.getString();     //Resposta payload
121
122    // Status de resposta. Exemplos: 200 --> OK, 404,-1 --> ERRO
123    Serial.println("Status da conexão = " + httpCode);
124
125    http.end();  //Fechando a conexão
126  } else {
127    Serial.println("Error in WiFi connection");
128  }
```

Fonte: Elaborado pelo autor.

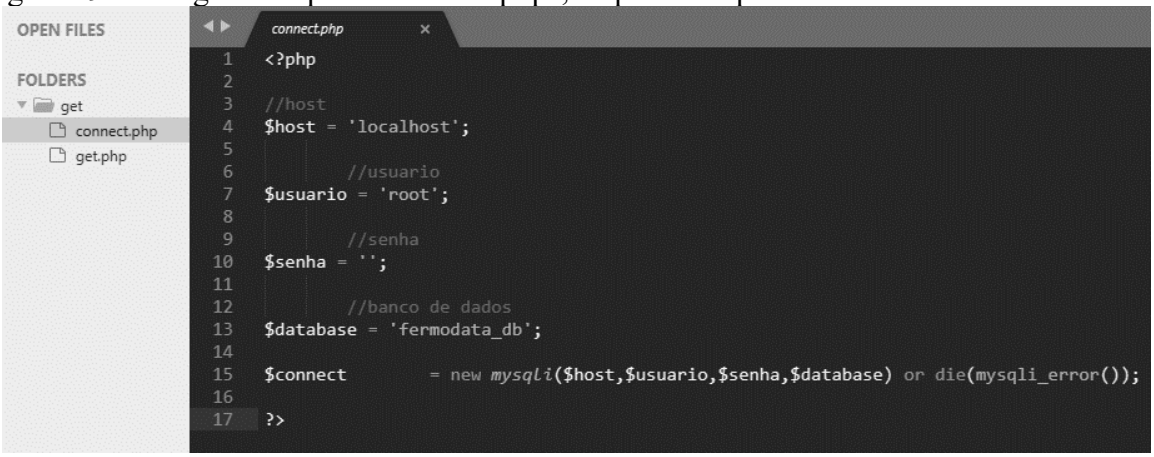
Conforme o código da Figura 28, primeiramente é validado se o microcontrolador está conectado ao Wi-Fi, em seguida é gerado a URL, indicada pela seta, com os valores dos sensores inseridos por parâmetro e, por fim, a requisição é enviada ao servidor *web*. A resposta, que é irrelevante para o microcontrolador neste projeto, é apenas armazenada em uma variável

local para visualização do *status* da conexão, que pode ser respostas positivas de confirmação (sucesso no recebimento, entre 200 a 299) ou respostas negativas (erros, entre 400 a 599).

3.4.3.4. Lado do Servidor

Do lado do servidor, existem dois arquivos PHP responsáveis por receber e enviar os dados provindos do microcontrolador para o banco de dados MySQL. Um é responsável por realizar a conexão com o banco de dados e o outro fica responsável por capturar a requisição GET e enviar ao banco de dados os parâmetros da URL recebida (ISRAEL, 2019).

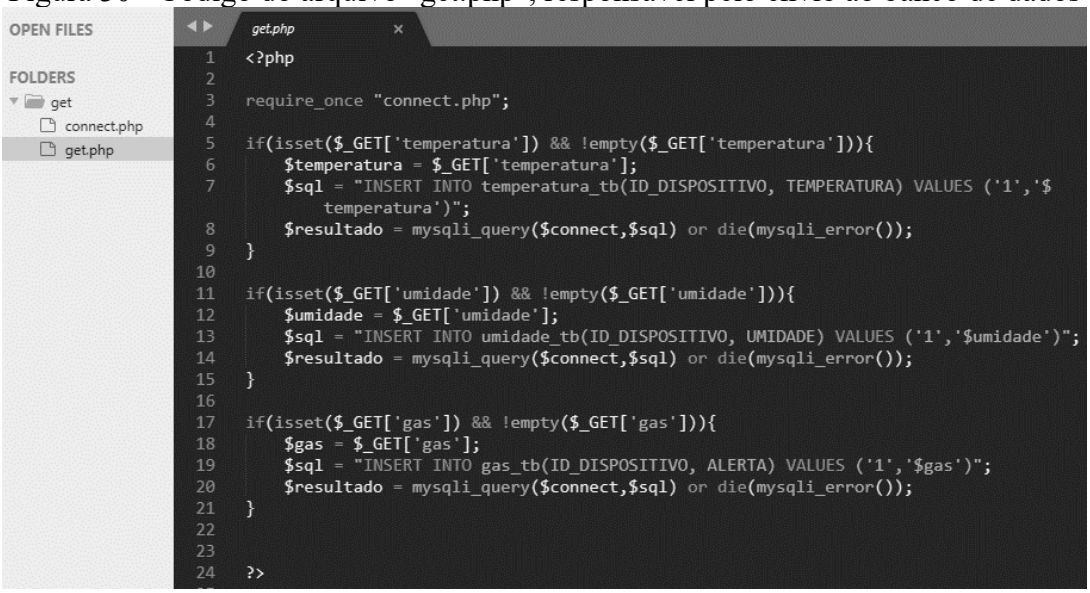
Figura 29 - Código do arquivo "connect.php", responsável pela conexão com banco de dados



Fonte: Elaborado pelo autor.

Como mostra o código da Figura 29, o arquivo “connect.php” realiza a conexão com o banco de dados “fermodata_db”, através de parâmetros de acesso, como o local do servidor (*host*), usuário e senha.

Figura 30 - Código do arquivo "get.php", responsável pelo envio ao banco de dados



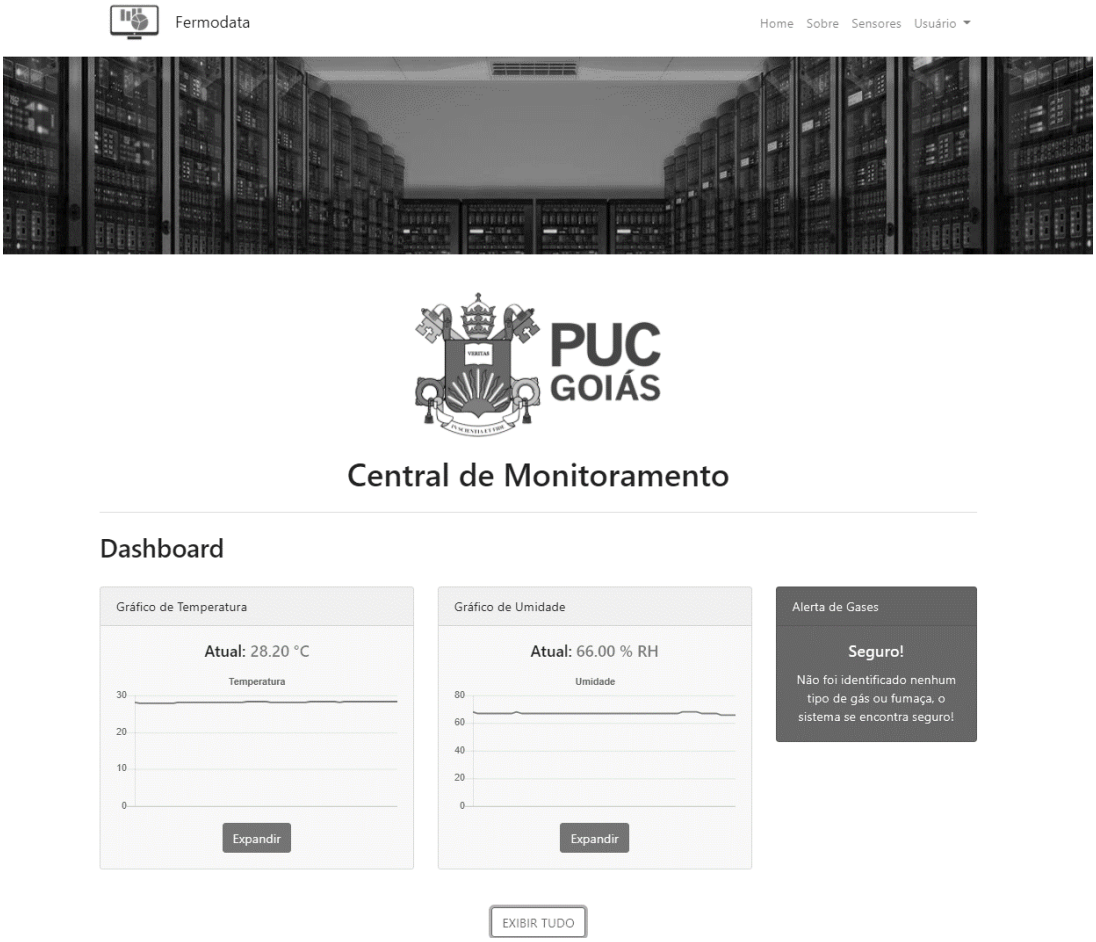
Fonte: Elaborado pelo autor.

O arquivo “get.php” (Figura 30), que recebe a requisição GET do microcontrolador, estabelece a conexão com o banco de dados, interpreta os valores recebidos por parâmetro da requisição, armazena os valores em variáveis do tipo temperatura, umidade e gás e as envia ao banco de dados através de comandos em SQL do tipo “INSERT INTO”, que insere os valores dentro das tabelas do MySQL.

3.5. Plataforma *Web*

A Plataforma *Web* é uma importante ferramenta de monitoramento, através dela é feita a visualização dos dados presentes do banco de dados de forma mais simples e intuitiva, através de gráficos e tabelas. Pode ser acessada de qualquer navegador em qualquer lugar desde que o endereço de IP do servidor esteja exposto ou visível para a WAN (*Wide Area Network* - Rede de Longa Distância, ou *Internet* pública).

Figura 31 - *Interface* da plataforma *web* de monitoramento



Fonte: Elaborado pelo autor.

A Figura 31, apresenta a *interface* da página de monitoramento do *Data Centers*. Nela estão disponíveis os dados retirados em tempo real do microcontrolador, dados estes que já estão armazenados no banco de dados e estão sendo capturados e exibidos pela plataforma *web* na forma de gráficos e tabelas. Nas próximas seções será trabalhado de forma mais detalhada as etapas de construção da página e alguns elementos que compõem sua visualização.

3.5.1. Hospedagem no Servidor Web

Antes de tudo, para ter acesso a plataforma *web* de monitoramento (*web*site) é necessário que os dados estejam hospedados em algum servidor *web*. O servidor *web* é responsável por receber as requisições do lado do cliente (dos navegadores) e responder de acordo com o que foi pedido, enviando para o outro lado os dados da página solicitada. No caso do servidor *web*, essa comunicação com o cliente é feita utilizando o protocolo HTTP, que é um protocolo que estabelece diretrizes de comunicação entre os dispositivos clientes e

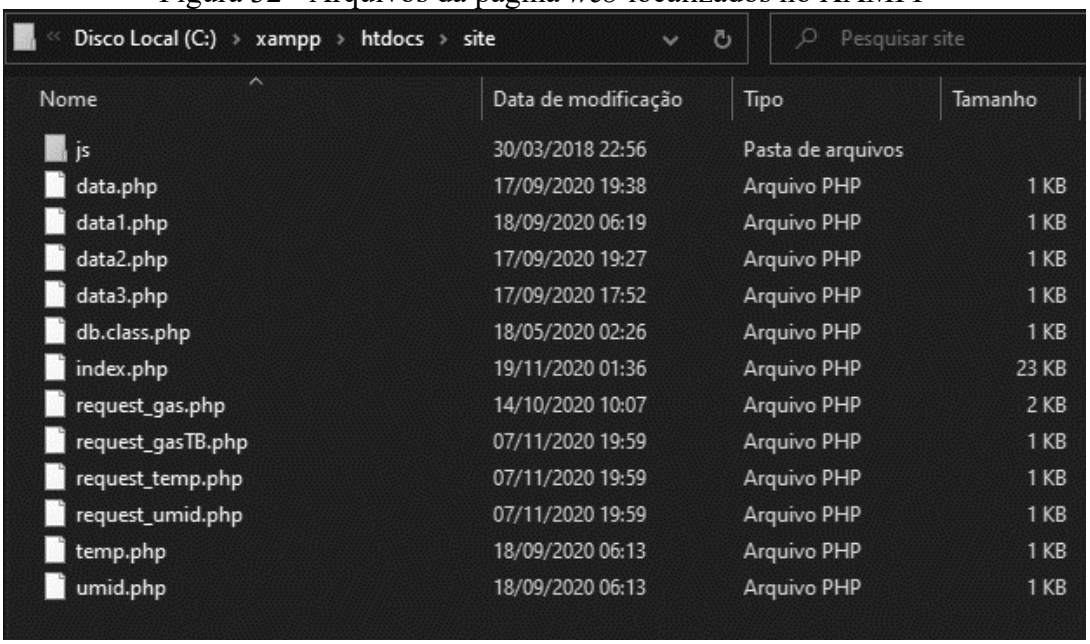
servidores, sendo o mesmo protocolo já trabalhado na sessão 3.4.3.2 desta monografia, para receber os dados do microcontrolador (TAVARES, 2019).

Como mencionado também na seção 3.4.3. desta monografia, através do *software* XAMPP, que disponibiliza uma série de servidores de código abertos e gratuitos, é criado o servidor *web* Apache, no qual está armazenado todos os arquivos da página de monitoramento, ou "Fermodata", nome do qual recebe a página de monitoramento de *Data Center* deste projeto de monografia.

O servidor *web* Apache, também conhecido como "Apache HTTP *Server*", disponibilizado pela Apache *Software* Foundation, é um *software* de código aberto que deve ser executado em um servidor físico, no qual, estejam armazenados os dados das páginas que ele deve exibir ao cliente solicitante através do protocolo de comunicação HTTP. De acordo com a própria desenvolvedora, o Apache é utilizado por grande parte dos servidores físicos que alimentam a *Internet* atualmente, sendo o mais popular desde abril de 1996 (APACHE,2020; LONGEN, 2020).

Neste projeto de monografia, a execução do Apache e o gerenciamento do servidor é feito em uma máquina na própria rede local. Os arquivos da página de monitoramento estão armazenados dentro da máquina que está executando o *software* XAMPP, o Apache e o MySQL, ao mesmo tempo, e podem ser acessados e visualizados apenas por dispositivos conectados na mesma rede.

Figura 32 - Arquivos da página *web* localizados no XAMPP



Fonte: Elaborado pelo autor.

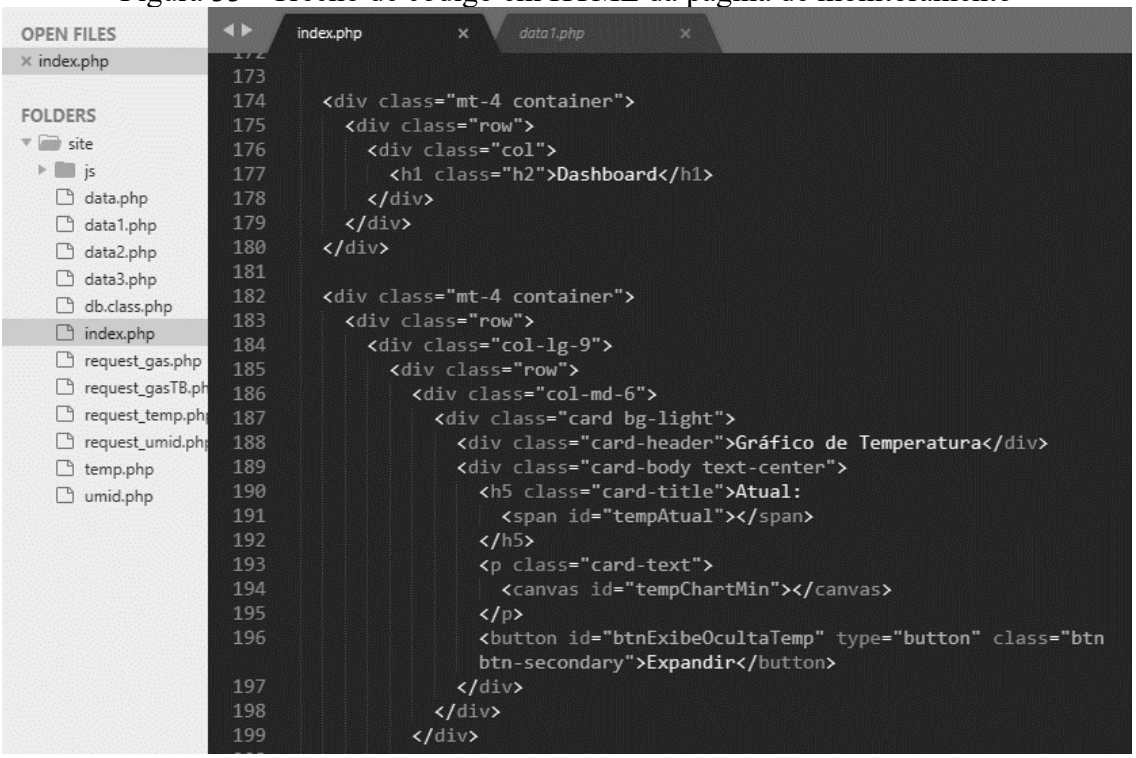
Como mostra a Figura 32, todos os arquivos do servidor *web* estão no formato ".php". Isso ocorre justamente, porque através das funções disponibilizadas pela linguagem PHP foi possível realizar a comunicação com o banco de dados MySQL. Além disso, a linguagem nos permite criar páginas em HTML e embutir códigos em PHP em sua estrutura. Dentre todos os arquivos da Figura 32, vale destacar o "index.php", que se trata da página principal da plataforma *web*, onde todas as informações vão se reunir para serem exibidas ao usuário (PHP, 2020; SANT; DAMASCENO, 2017).

3.5.2. Página de Monitoramento

A plataforma *web* de monitoramento do *Data Center*, como mostrada na Figura 31, consiste em uma única página de *dashboard* (painel de indicadores), com diversos elementos que exibem de formas diferentes os três principais dados: temperatura, umidade e alerta de gás. A demonstração é feita através de gráficos, *cards* e tabelas. Toda a estrutura foi montada utilizando HTML5, *Bootstrap* e JavaScript, além do PHP, que é o responsável pela comunicação com o banco de dados.

O HTML (*Hypertext Markup Language* – Linguagem de Marcação de Hipertexto), utilizada para o desenvolvimento de *websites*, permite construir as estruturas básicas de uma página *web*. Por ser uma linguagem de marcação, para cada elemento (também chamados de Hipertextos) que compõe a página, deve existir uma marcação para informar ao navegador a forma e quais as informações que deverão ser exibidas na página. O HTML é uma linguagem interpretada pelos navegadores (Chrome, Firefox, Opera, Edge dentre outros) e já se encontra na sua quinta versão, por isso HTML5 (MARQUES, 2019).

Figura 33 - Trecho de código em HTML da página de monitoramento



Fonte: Elaborado pelo autor.

Na Figura 33, é mostrado um trecho em HTML da página *web* de monitoramento. Todo texto e elementos que aparecem são marcados através de "*tags*", que indicam ao interpretador do navegador como as partes deverão ser exibidas.

Juntamente com o HTML, que é responsável pelo conteúdo que aparece na página, também é utilizado na codificação da página o *Bootstrap*, que nada mais é que um *framework* com uma coleção de sintaxes responsáveis pela estilização e *layout* da página. O *framework* permite a criação de páginas responsivas a tela de qualquer dispositivo e possui diversos recursos otimizados que são bastante utilizados na página *web*, como botões, caixas de textos, *cards*, tabelas, barras de navegação, dentre outros (BOOTSTRAP, 2020).

Geralmente quem assume esse papel de estilização das páginas em HTML é o CSS (*Cascading Style Sheets* - Folha de Estilo em Cascatas). O *Bootstrap* é a junção de diversos códigos de estilo em CSS reunidos em um trecho de código só, criando algo como sua própria biblioteca de estilos e *layout*, otimizando o tempo de programação do desenvolvedor (GONÇALVES, 2019).

Para o bom funcionamento do *Bootstrap*, e para manter a dinâmica da página é utilizado também o *JavaScript*, que é o responsável por gerar a interatividade e o movimento da página, sendo uma linguagem de programação inserida em *scripts* dentro da página HTML. No

monitoramento do *Data Center*, o *JavaScript* é o grande responsável por atualizar informações, gráficos e esconder e exibir elementos em tempo real na página (BOOTSTRAP, 2020).

Para auxiliar no uso do *JavaScript* na página de monitoramento deste projeto de monografia são utilizadas duas grandes bibliotecas de *JavaScript*: *jQuery* e a *Chart*.js. A biblioteca *jQuery* auxilia na utilização de funções Ajax (*Asynchronous JavaScript and XML* - Javascript Assíncrono e XML), responsáveis por atualizar informações na página de forma dinâmica. Já a biblioteca *Chart*.js é uma biblioteca usada para a criação de diversos tipos de gráficos, que juntamente com o Ajax é capaz de produzir gráficos dinâmicos.

3.5.3. Dados em Tempo Real

Para exibir os dados na página em tempo real e de forma dinâmica é utilizado uma função denominada Ajax, que juntamente com a biblioteca jQuery e a linguagem PHP, realizam a comunicação com o banco de dados e atualizam os dados exibidos para o usuário (SANT; DAMASCENO, 2017).

3.5.3.1. Exibição dos dados com Ajax

O Ajax, é uma combinação de técnicas que utilizam *JavaScript*, XML (*eXtensible Markup Language* - Linguagem de Marcação Extensível) e HTML, utilizado para efetuar o carregamento de informações de forma assíncronas. O que significa que de tempos em tempos o Ajax envia requisições para o servidor de bancos de dados solicitando informações para serem atualizadas na página *web* sem a necessidade de o usuário ficar recarregando a página. O Ajax recebe esse nome pois usa o *JavaScript* na parte de gerenciamento do conteúdo dinâmico e o XML para armazenar e transmitir os dados no estilo do HTML (FARIAS, 2012; SANT; DAMASCENO, 2017).

O *jQuery* é uma compactação de diversas linhas de códigos em *JavaScript*, resumindo grandes trechos de códigos em pequenas funções prontas. Ele funciona juntamente com o Ajax, tornando-o mais simples e funcional, adaptando o código Ajax a todos os tipos de navegadores (SOUZA,2020).

Figura 34 - Trecho de código utilizando a função Ajax, para atualização dos valores de temperatura

```
71  function ajx_tempAtual(){
72      var ajax = new XMLHttpRequest(); // cria o objeto XHR
73      ajax.onreadystatechange = function(){
74          // verifica quando o Ajax for completado
75          if(ajax.readyState == 4 && ajax.status == 200){
76              document.getElementById("tempAtual").innerHTML = ajax.responseText; //
                atualiza o span
77              setTimeout(ajx_tempAtual, 3000); // chama a função novamente após 2 segundos
78          }
79      }
80      ajax.open("GET", "temp.php"); // página a ser requisitada
81      ajax.send(); // envia a requisição
82  }
83  ajx_tempAtual(); // chama a função
```

Fonte: Elaborado pelo autor.

Na Figura 34, é demonstrado um trecho de código em que foi utilizado o Ajax para realizar a exibição da última captura de temperatura inserida no banco de dados pelo microcontrolador. A função Ajax, faz uma requisição ao arquivo “temp.php” (linha 80), e este faz comunicação com o banco de dados e retornar a temperatura desejada. Em seguida, a função insere a resposta obtida em um elemento HTML da página principal. Lembrando que o processo se repete para a obtenção do demais valores de umidade e gás.

Figura 35 - Trecho em HTML, que recebe os valores atualizados da função Ajax

```
190                 <h5 class="card-title">Atual:
191                   <span id="tempAtual"></span>
192                 </h5>
```

Fonte: Elaborado pelo autor.

Figura 36 - Resulta da exibição do elemento na página de monitoramento

Fonte: Elaborado pelo autor.

O elemento HTML que está recebendo o valor atualizado do Ajax é identificado entre as *tags* “”, presente na linha 191 da Figura 35, que nada mais é que um bloco de texto genérico. Mas o que realmente vale destacar neste trecho é o “id”, que é a identificação do elemento, sendo essa identificação a responsável receber as atualizações da função Ajax, vista na linha 76 da Figura 34. O resultado da exibição também pode ser visto na Figura 36.

3.5.3.2. Exibição dos dados com PHP

Como já mencionado na seção 3.4.2. desta monografia, a linguagem PHP é bastante utilizada em páginas *web* para realizar comunicações com o lado do servidor. O PHP possui funções nativas próprias para se conectar com o banco de dados e realizar consultas (ESTRELLA, 2020).

Para realizar a conexão com o banco de dados e realizar as consultas são utilizados dois arquivos diferentes. O primeiro arquivo denominado “db.class.php” é utilizado sempre que necessário realizar a conexão com o banco de dados, e uma vez que a conexão esteja estabelecida, o segundo arquivo entra em ação. No caso, o segundo é utilizado para realizar as consultas ao banco de dados, sendo diferentes arquivos para diferentes consultas (SANT; DAMASCENO, 2017).

Figura 37 - Arquivo "db.class.php" utilizado para a conexão com o banco de dados

db.class.php

```php
<?php
    class db {
        //host
        private $host = 'localhost';
        //usuario
        private $usuario = 'root';
        //senha
        private $senha = '';
        //banco de dados
        private $database = 'fermodata_db';

        public function conecta_mysql(){

            //criar conexao
            $con = mysqli_connect($this->host, $this->usuario, $this
                ->senha, $this->database);

            //ajustar o charset de comunicação entre a aplicação e o
            banco de dados
            mysqli_set_charset($con, 'utf8');

            //verificar se houve erro de conexão
            if(mysqli_connect_errno()){
                echo 'Erro ao tentar se conectar com BD MySQL: ' .
                    mysqli_connect_error();
            }
            else{
            }
            return $con;
        }
    }
?>
```

Fonte: Elaborado pelo autor.

Figura 38 - Arquivo "temp.php" utilizado para realizar a consulta ao banco de dados

temp.php

```php
<?php

require_once('db.class.php');

$sql = "SELECT * FROM temperatura_tb ORDER BY ID DESC LIMIT 1";

$objDb = new db();
$link = $objDb->conecta_mysql();

$resultado_id = mysqli_query($link, $sql);

if($resultado_id){

  while ($dados_usuario = mysqli_fetch_array($resultado_id)) {
    $temp = $dados_usuario['TEMPERATURA'];
    echo '<span style="color:#00c0ef">'.$temp." ºC</span>";
}



}else{

  echo "Erro na execução da consulta, favor entrar em contato com o
      admin do site";

}
?>
```

Fonte: Elaborado pelo autor.

Continuando o exemplo da seção anterior, de consulta da temperatura, o arquivo “temp.php” requisitado pela função Ajax (linha 80, Figura 34) vai realizar a consulta no banco de dados e retornar o último valor de temperatura registrado.

O código da Figura 38, do arquivo “temp.php” segue os seguintes passos:

1. Realiza a conexão com o banco de dados MySQL, através do arquivo “db.class.php”;
2. Faz a consulta do último valor inserido na tabela de temperatura (“temperatura_tb”);
3. Armazena o resultado obtido em uma variável denominada “$temp”;
4. Através do comando de saída do PHP, o “echo”, é exibido o valor na página.

O Ajax realiza a requisição a esse arquivo diversas vezes em segundo plano, exibindo a saída “echo” do arquivo PHP em cada uma delas.

3.5.4. Gráficos

Os gráficos são elementos essenciais para uma plataforma de monitoramento de *Data Centers*, através deles são exibidos os elementos de temperatura e umidade ao longo do tempo,

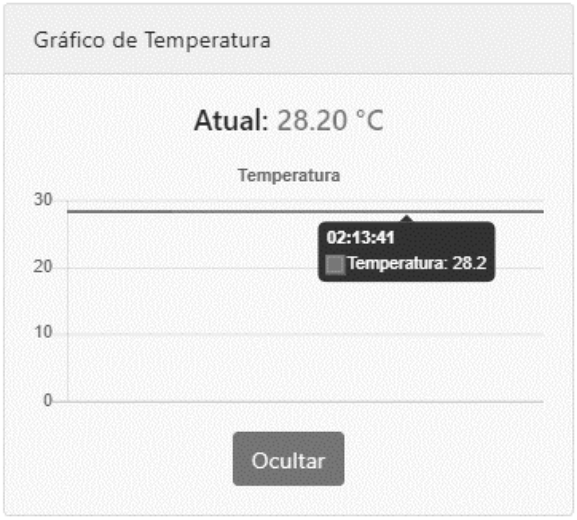
nos ajudando a ter uma melhor interpretação dos dados, visualizando tendências e discrepâncias que podem ser percebidas ao longo do tempo (ALGAR, 2017).

Para não perder tempo de reação, principalmente em uma central de monitoramento de um *Data Center*, é muito importante que os gráficos sejam atualizados a cada momento de forma dinâmica e sem a necessidade de o usuário ficar recarregando a página. Para isso, foram utilizados o *JavaScript*, Ajax e o Chart.js (*WEB*SLESSON, 2018).

O *Chart*.js nada mais é que uma biblioteca em JavaScript cujo foco principal é criar gráficos de todos os tipos. Para renderizar os gráficos, a biblioteca também fazem o uso de elementos em HTML e CSS, utilizando-se do elemento “canvas” do HTML5. A biblioteca disponibiliza gráficos do tipo em barras, pizza, linha e diversos outros (LACROIX, 2018).

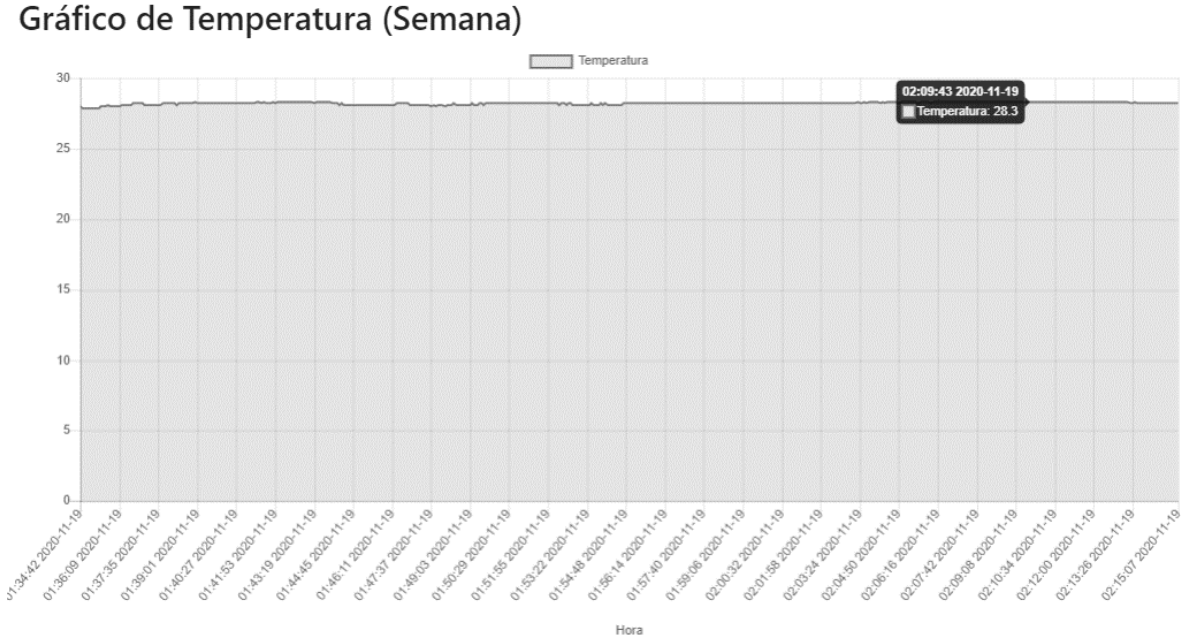
Na plataforma *web* de monitoramento de *Data Centers* são utilizados gráficos em linha para a representação de temperatura e umidade ao longo do tempo. As unidades de tempo são representadas no eixo horizontal e os valores variáveis, de temperatura e umidade, são representados no eixo vertical do gráfico. Os gráficos são responsivos e dinâmicos, possuem legendas com informações de valores, data e hora, como pode ser visto nas Figuras 39 e 40.

Figura 39 - Gráfico de temperatura dos últimos 5 minutos



Fonte: Elaborado pelo autor.

Figura 40 - Gráfico de temperatura da semana



Fonte: Elaborado pelo autor.

O *Dashboard* da plataforma possui duas representações de gráficos, um gráfico menor, Figura 39, para representar os valores de temperatura e umidade com valores mais recentes, com medições dos últimos 5 minutos, presente dentro de um "*card*" (miniatura com container flexível), e outro gráfico maior, Figura 40, que demonstra as medições da última semana. Esses valores foram escolhidos apenas para fins de testes, os gráficos e medições podem ser exibidas desde o mês completo até dos últimos anos, ficando a critério da necessidade do projeto.

A criação dos gráficos e a renderização em tempo real é feita em basicamente em três etapas. A primeira etapa consiste em definir os parâmetros dos gráficos, com suas características e estilos, como demonstrado na Figura 41.

Figura 41 - Código utilizado para definir os parâmetros do gráfico de temperatura

```
546  <script>
547  // create initial empty chart
548  var ctx_live = document.getElementById("tempChartMin");
549  var myChartTempMin = new Chart(ctx_live, {
550    type: 'line',
551    data: {
552      labels: [],
553      datasets: [{
554        data: [],
555        borderWidth: 3,
556        borderColor: '#00c0ef',
557        backgroundColor: '#00c0ef',
558        label: 'Temperatura',
559        fill: false ,
560        pointRadius: 0,
561        fill: false,
562        lineTension: 0,
563        borderWidth: 2}] },
564        options: {
565          animation: {
566            duration: 0
567          },
568          responsive: true,
569          title: {
570            display: true,
571            text: "Temperatura"
572          },
573          tooltips: {
574            mode: 'index',
575            intersect: false,
576          },
577          hover: {
578            mode: 'nearest',
579            intersect: true
580          },
581          legend: {
582            display: false
583          },
584
585          scales: {
586            xAxes: [{
```

Fonte: Elaborado pelo autor.

A segunda etapa consiste na utilização da função Ajax, cujo processo de implementação é bem semelhante ao trabalhado na seção 3.5.2.1. desta monografia. Após a solicitação ao arquivo PHP, os valores são inseridos e atualizados no gráfico a cada 3 segundos, como mostra a Figura 42 (linha 620 e linha 625 respectivamente).

Figura 42 - Função Ajax para atualização do gráfico de temperatura

```
var getData = function () {
  $.ajax({
    url: 'https://jsonplaceholder.typicode.com/posts/' + postId
      + '/comments',
    success: function (data) {
      $.post("data2.php",
        function (data)
        {
          var temps = [];
          for (var i in data) {
            myChartTempMin.data.labels[i] = data[i].HORA;
            myChartTempMin.data.datasets[0].data[i] = data[i].
              TEMPERATURA;
          }

          myChartTempMin.update();
        });
    } });
};
// Atualiza a cada 3 secondos
setInterval(getData, 3000);
```

Fonte: Elaborado pelo autor.

Na última etapa de criação do gráfico, é feita apenas a inserção do gráfico dentro do elemento HTML, no corpo principal da página. Essa inserção é feita entre as *tags* “<canvas>” presente no HTML5, própria para renderização de elementos gráficos. O elemento também deve possuir uma identificação “id”, para receber o gráfico gerado.

Figura 43 - Trecho de código em HTML que recebe o gráfico já atualizado da função Ajax

```
<p class="card-text">
  <canvas id="tempChartMin"></canvas>
</p>
```

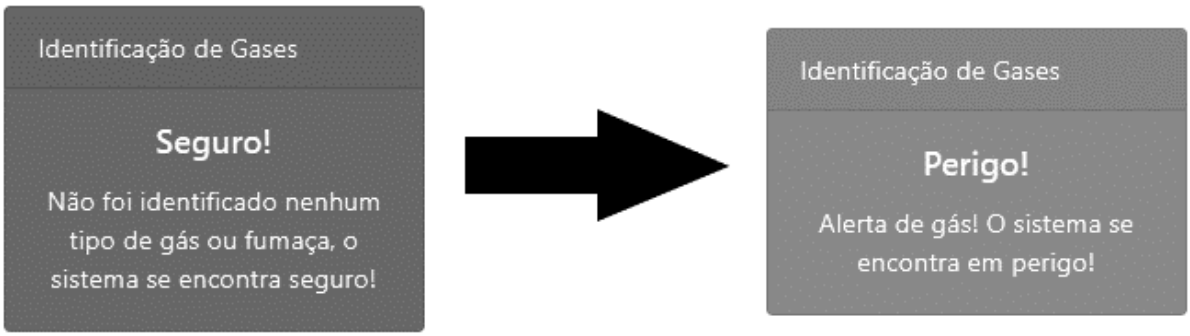
Fonte: Elaborado pelo autor.

3.5.5. Alerta de Gás e Fumaça

O alerta de gás funciona através de respostas binárias provindas do sensor de gás, respostas estas que são representadas dentro do banco de dados com os caracteres “S”, para detecção de gás e “N”, para a ausência de gás.

A identificação do alerta de gás dentro da plataforma *web* de monitoramento é feita através de um “card” ou cartão, que é uma miniatura em forma de container que inclui bordas, cabeçalho e rodapé. O elemento trata-se de uma estilização de divisória criada pelo *Bootstrap*. Ele alterna entre as cores verde e vermelho representando ausência de alertas e detecção de gases inflamáveis e fumaça respectivamente, como mostra a Figura 44.

Figura 44 - Estados de alerta de gás



Fonte: Elaborado pelo autor.

Da mesma forma que os demais elementos, o alarme identificador de gás também realiza consultas no banco de dados em tempo real através da função Ajax e a linguagem PHP, como já relatado na seção 3.5.2.1. desta monografia.

3.5.6. Tabelas de Consulta

A tabela de consulta é responsável por exibir todos os valores presentes nos bancos de dados, sem filtro algum. O uso desta tabela em plataformas de monitoramento real não é muito indicado, visto que sua visualização não é de fácil análise e dependendo do tamanho da tabela pode gerar lentidão para o carregamento de todos os dados na página. Ela se encontra presente nesta plataforma apenas para demonstrar todo o histórico de coleta realizado ao longo do tempo para fins de testes e análise.

Figura 45 – Tabela com valores de umidade, gás e temperatura

Tabela de Consulta

Umidade		
%	DATA	HORA
68.00	2020-11-19	01:34:42
67.00	2020-11-19	01:34:47
67.00	2020-11-19	01:34:53
67.00	2020-11-19	01:34:58
67.00	2020-11-19	01:35:03
67.00	2020-11-19	01:35:08
67.00	2020-11-19	01:35:13
67.00	2020-11-19	01:35:18
67.00	2020-11-19	01:35:23

Gás		
ALERTA S/N	DATA	HORA
N	2020-11-19	01:34:42
N	2020-11-19	01:34:47
N	2020-11-19	01:34:53
N	2020-11-19	01:34:58
S	2020-11-19	01:35:03
S	2020-11-19	01:35:08
S	2020-11-19	01:35:13
S	2020-11-19	01:35:18
S	2020-11-19	01:35:23

Temperatura		
°C	DATA	HORA
28.00	2020-11-19	01:34:42
27.90	2020-11-19	01:34:47
27.90	2020-11-19	01:34:53
27.90	2020-11-19	01:34:58
27.90	2020-11-19	01:35:03
27.90	2020-11-19	01:35:08
27.90	2020-11-19	01:35:13
27.90	2020-11-19	01:35:18
27.90	2020-11-19	01:35:23

Fonte: Elaborado pelo autor.

As tabelas são construídas de forma manual, como mostra a Figura 46 (trecho de código utilizado para criação da tabela de umidade), inserindo e retirando bordas nas divisórias, que são os elementos HTML identificados entre as *tags* “ ”. Da mesma forma que nas seções 3.5.3, 3.5.4 e 3.5.6, seus valores são atualizados através da função Ajax, dentro do elemento identificado por “id=umidade_exibicao”. As tabelas também são atualizadas a cada 3 segundos.

Figura 46 - Trecho de código em HTML utilizado para criação da tabela de umidade

```
269     <div class="mt-3 container border-top mb-5" >
270       <div class="mt-3 row">
271         <div class="col-md-4 border-right pl-3 pr-3">
272           <h3 class="text-center" style="color:#00c0ef">Umidade</
              h3>
273           <div class="row mt-4 ml-auto mr-auto">
274             <div class="col p-2 font-weight-bold border
              border-secondary border-bottom-0 text-center"style="
                color:#00c0ef">
275               %
276             </div>
277             <div class="col p-2 font-weight-bold border
              border-secondary border-bottom-0 border-left-0
              text-center">
278               DATA
279             </div>
280             <div class="col p-2 font-weight-bold border
              border-secondary border-bottom-0 border-left-0
              text-center">
281               HORA
282             </div>
283           </div>
284           <div id="umidade_exibicao"></div>
285           <div class="row ml-auto mr-auto">
286             <div class="col border border-secondary
              border-bottom-0 border-left-0 border-right-0">
287             </div>
288           </div>
289         </div>
```

Fonte: Elaborado pelo auto.

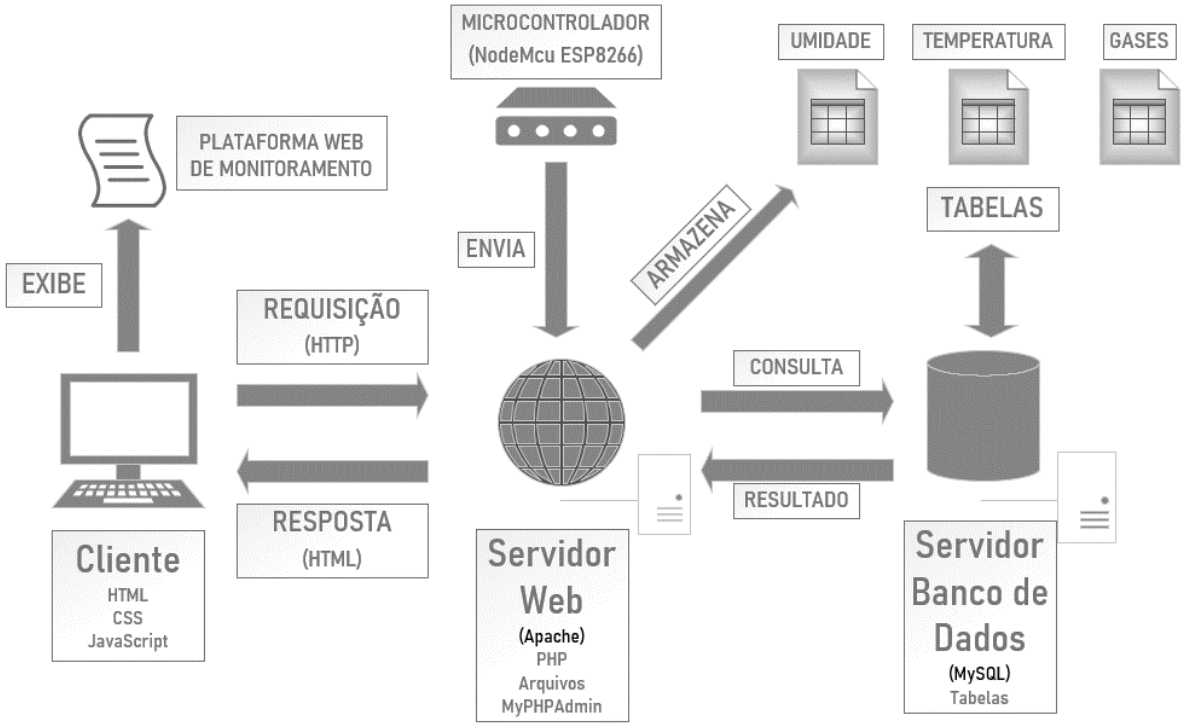
CAPÍTULO 4 RESULTADOS E TESTES

Após a montagem do modelo representativo do *Data Center* (maquete), criação e implementação do microcontrolador, criação do banco de dados e da plataforma *web* de monitoramento tem-se a conclusão de todas etapas de criação de uma central de monitoramento de um *Data Center*. A partir deste momento, o projeto está apto para a realização de testes e documentação dos resultados obtidos.

Antes de partir para os testes em si, é importante revisar todas as etapas de funcionamento da central de monitoramento do *Data Center*:

- Captura do estado climático do ambiente através de sensores;
- Os sensores enviam os dados coletados para o microcontrolador;
- O microcontrolador envia os dados para o servidor *web* que repassa para o banco de dados;
- A plataforma *web* de monitoramento exibe os dados armazenados no banco de dados.

Figura 47 - Diagrama com fluxo de funcionamento da central de monitoramento



Fonte: Elaborado pelo autor.

Todas as etapas trabalham em paralelo, transmitindo as informações em tempo real com um atraso de no máximo de 5 segundos. Gerando assim, um tempo de resposta relativamente baixo o suficiente para tomar ações de acordo com o que é exibido em tela.

4.1 Ambiente de Testes

O ambiente onde os testes foram realizados trata-se de uma maquete, que é um modelo representativo de um *Data Center* em escala menor. A maquete é utilizada apenas para ter uma noção básica do posicionamento dos elementos dos dispositivos principais de um *Data Center* real.

O microcontrolador é posicionado em um compartimento interno acima dos *racks.* Na maquete está sendo representado pelo cabeamento estruturado elevado, como mostra na Figura 48.

Figura 48 - Compartimento onde se encontra o microcontrolado na maquete

Fonte: Elaborado pelo autor.

Neste mesmo compartimento existem saídas laterais para o posicionamento dos sensores de temperatura, umidade e gases, localizados na parte da frente e de trás da maquete

respectivamente. A frente da maquete representa a parte frontal dos *Racks*, onde é posicionado o sensor de temperatura e umidade. Já a parte de trás dos *Racks*, é o local em que está posicionado o sensor de gás.

4.1.1 Temperatura e Umidade

O sensor de umidade e temperatura, posicionado na parte frontal dos *racks*, representa o local onde fica localizada a frente dos equipamentos posicionados no *rack* de um *Data Center* real. O local se torna mais propício para o posicionamento do sensor de temperatura e umidade pois é o local mais indicado para as saídas do ar frio do ar-condicionado. Então, neste modelo representativo, é considerado que o ar frio entre pela frente dos equipamentos.

Assim, considerando que haja qualquer falha no ar-condicionado ou no próprio equipamento presente no *Rack*, será identificado de forma mais rápida e eficiente se o sensor de temperatura e umidade estiver posicionado na parte frontal dos equipamentos, facilitando a identificar a anomalia presente no ambiente.

O projeto de monitoramento desta monografia visa gerar um sistema responsivo, que seja capaz de identificar e exibir ao usuário problemas relacionados a mudanças no ambiente de um *Data Center*. Assim, para melhorar entendimento e facilitar a visualização e compreensão de mudanças, os testes deste projeto foram feitos utilizando utensílios domésticos para forçar mudanças drásticas no ambiente físico da maquete.

Figura 49 - Processo de aquecimento com secador de cabelo

Fonte: Elaborado pelo autor.

Para a realização dos testes de temperatura e umidade é necessário gerar um grande aumento ou queda na temperatura e na umidade. Para isso foi utilizado um secador de cabelo, que é capaz de gerar ar quente para aumentar a temperatura e ao mesmo tempo é capaz de reduzir a umidade do ambiente.

4.1.2 Gases e Fumaça

O sensor de gás MQ-2 utilizado neste projeto é capaz de detectar fumaça e gases inflamáveis como o metano, propano, butano, hidrogênio, álcool, gás natural e outros (informações da própria fabricante). O sensor de gás se encontra posicionado na parte traseira dos equipamentos e dos *racks*, em vista que seja o local mais propenso ao aumento brusco de temperaturas e geração de incêndios provindo de falhas no equipamento ou curtos circuitos.

Figura 50 - Processo para liberação de gás butano com isqueiro

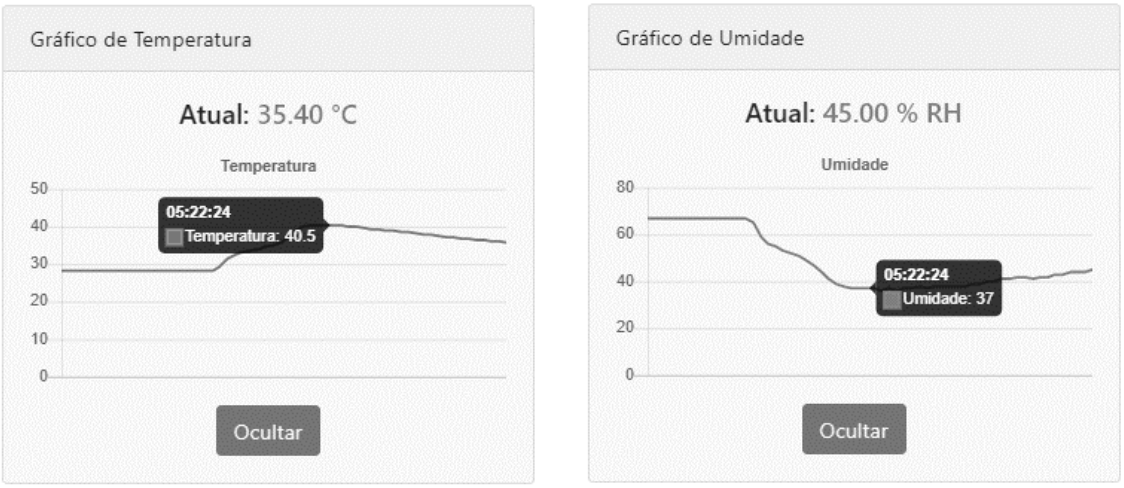
Fonte: Elaborado pelo autor.

Para a realização dos testes do sensor de gás MQ-2 é necessário gerar fumaça ou algum gás inflamável próximo do dispositivo. Assim devido ao perigo de expor a maquete e os demais dispositivos a fumaça, foi utilizado um isqueiro para a liberação de gás butano que é altamente inflamável, que em poucas quantidades já é possível gerar resultados para os testes.

4.2 Resultados

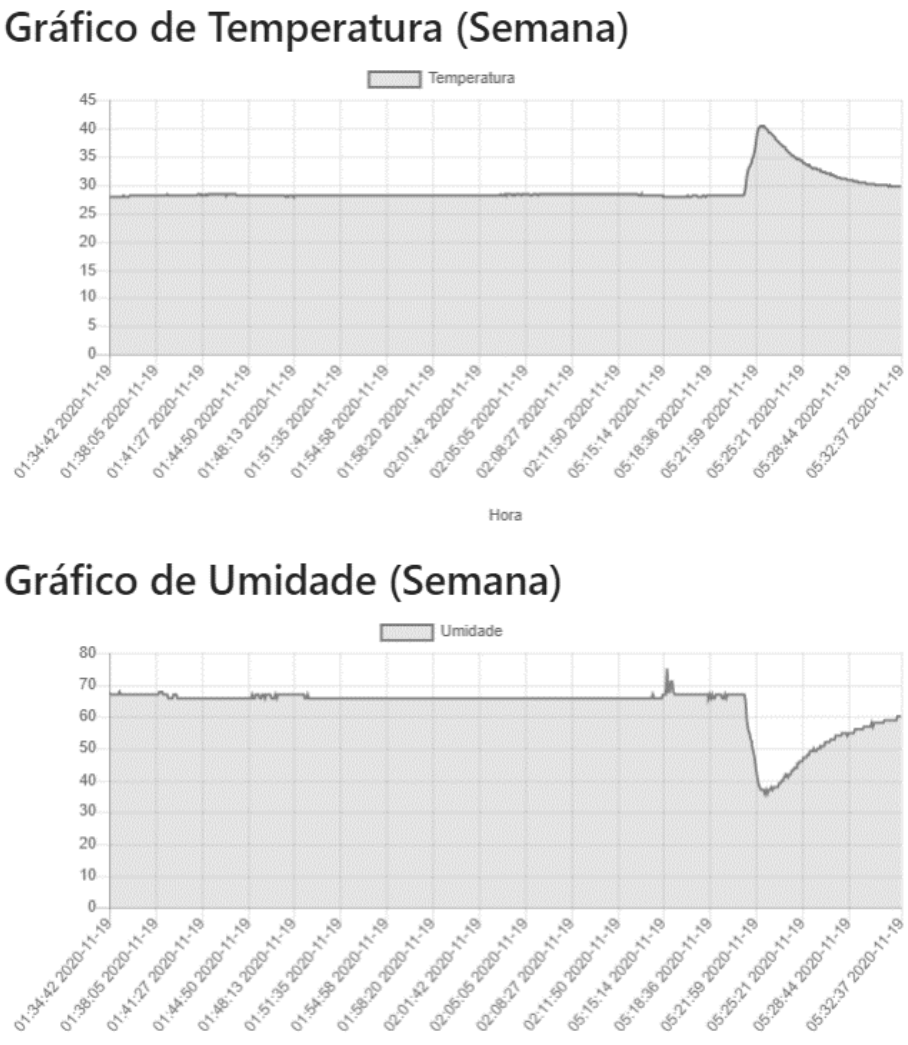
Após os testes de aquecimento com ar quente, rapidamente já se torna visível as mudanças nos gráficos de temperatura e umidade na plataforma *web* de monitoramento, como pode ser visto nas Figura 51 e 52.

Figura 51 - Mudanças após testes com aquecimento de temperatura, gráfico de 5 minutos



Fonte: Elaborado pelo autor.

Figura 52- Mudanças após testes com aquecimento de temperatura, gráfico da semana

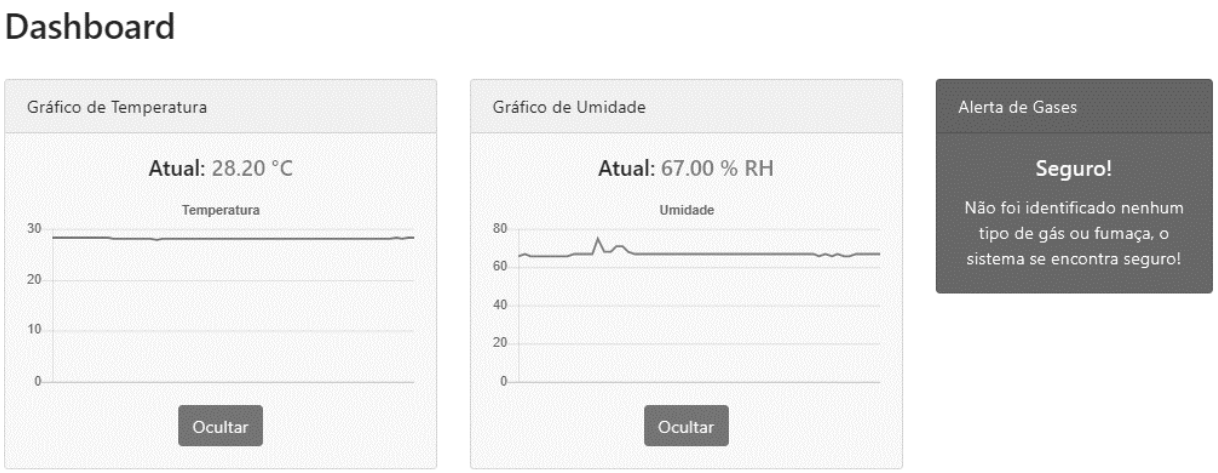


Fonte: Elaborado pelo autor.

Nas Figuras 51 e 52, a temperatura se mantinha estável em torno de 28.2 °C, após o teste com ar quente a temperatura aumentou bruscamente, atingindo um pico de 40.5 °C, e da mesma forma, houve uma queda de 67 % para 37 % da umidade relativa do ar. O teste foi efetuado com sucesso, demonstrando resultados satisfatórios.

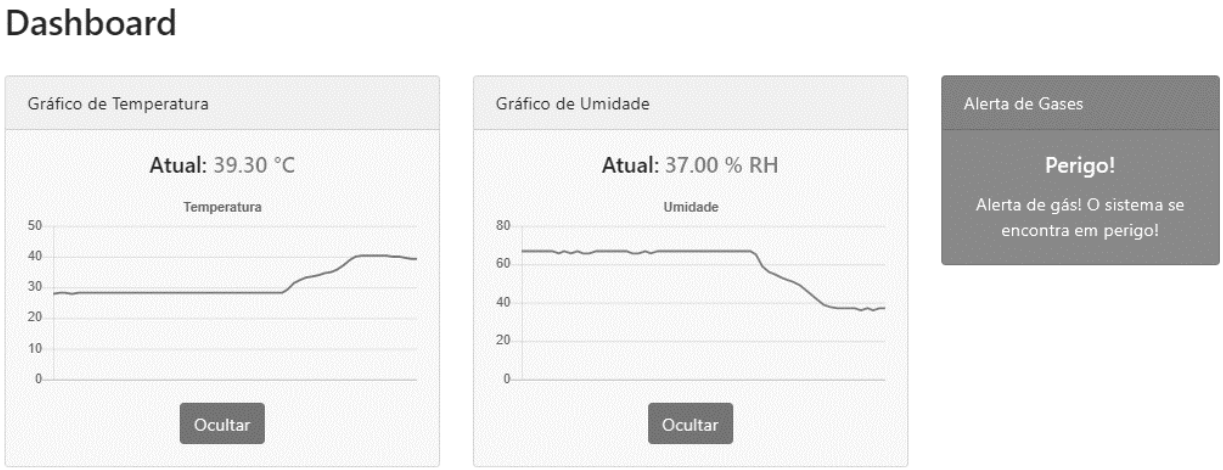
O teste com o sensor de gás, através da exposição de gás butano, também ocorreu com sucesso. Com a proximidade do gás ao sensor, ele rapidamente gera um sinal de alarme na plataforma *web* de monitoramento, alarmando a detecção de gases inflamáveis e solicitando ao usuário tomar as medidas corretivas rapidamente, como pode ser visto nas Figuras 53 e 54.

Figura 53 - Alerta antes dos testes



Fonte 1: Elaborado pelo autor.

Figura 54 - Alerta depois dos testes



Fonte 2: Elaborado pelo autor.

4.2.1 Gráficos e Alarmes

Os gráficos se provaram através dos testes ferramentas de monitoramento adequadas. As mudanças de temperatura e umidade se tornaram visivelmente bem evidentes, auxiliando na compreensão e na visualização do histórico de mudanças no ambiente físico ao longo do tempo, gerando estatísticas para análises futuras.

O alarme de gás também se provou bastante útil na plataforma de monitoramento, através de uma resposta rápida e de um bom destaque com cores fortes como verde e vermelho, o alarme rapidamente consegue passar a mensagem de urgência ao usuário da plataforma de monitoramento.

4.2.2 Plataforma Web

A plataforma no geral se mostrou bastante satisfatória para a proposta deste projeto de monografia. Apesar de possuir uma *interface* simples com poucas informações na tela, ela se provou eficiente na identificação de anomalias no ambiente físico após a realização dos testes. A plataforma de monitoramento por ser simples se torna mais intuitiva, o que acaba tornando mais evidente a identificação das mudanças que ocorre no ambiente e na plataforma.

Além de sua simplicidade e facilidade de compreensão, a principal vantagem desta plataforma *web* de monitoramento é que o seu acesso pode ser feito remotamente de qualquer dispositivo conectado na mesma rede ou através de outras redes dependendo da configuração desejada pelo usuário. Com isso é possível realizar o monitoramento do ambiente de qualquer lugar, apenas tendo acesso à *Internet* e a qualquer navegador.

4.2.3 Tempo de Resposta

Com os testes, a plataforma *web* apresentou tempos de respostas bastante aceitáveis, em torno de 5 segundos para todas as mudanças se tornarem visíveis. Os textos, gráficos e o alarme são atualizados de 3 em 3 segundos de acordo com as informações presentes no banco de dados, já o tempo que o microcontrolador envia os dados para o banco de dados é de 5 em 5 segundos. Logo, assim que o banco recebe uma nova atualização a plataforma *web* já repassa a informação visual para o usuário.

O tempo de resposta da plataforma se provou rápido o suficiente para a geração de gráficos e alertas sobre qualquer anomalia ou problemas relacionados a temperatura, umidade

e gases no ambiente físico de teste da maquete do *Data Center*, sendo também o tempo necessário para que o usuário se torne apto a tomar medidas corretivas, evitando acidentes.

CAPÍTULO 5 CONCLUSÃO

Com o desenvolvimento teórico e prático deste trabalho, pode-se concluir que com a criação de uma central de monitoramento que automatize a detecção de temperatura, umidade e gases em *Data Centers* em tempo real é essencial para que sejam tomadas ações preventivas e corretivas, promovendo a proteção correta dos equipamentos e da vida humana que possa estar presente no ambiente físico dos *Data Centers*.

O estudo teórico sobre a segurança dos *Data Centers* permitiu obter uma melhor compreensão do funcionamento e da organização dentro desse tipo de ambiente, focando principalmente na sua segurança física e nos seus principais pontos fracos. Com isso, surgiu a proposta deste projeto, que visa proporcionar meios de corrigir um dos pontos fracos mais agravantes deste tipo de infra estrutura, o estado climático do seu ambiente físico.

Os equipamentos presentes nos *Data Centers* são extremamente caros e sensíveis, por esse motivo eles precisam operar em temperaturas e umidades ideias, e possuir sistemas contra fumaça e incêndios. A ideia de implementação de uma central de monitoramento surgiu justamente da necessidade de visualizar e analisar o comportamento do estado climático dentro deste tipo de infra estrutura, buscando meios de corrigir quaisquer problemas e anomalias climáticas.

O modelo representativo do *Data Center* (maquete) permitiu reproduzir situações de mudanças climáticas reais, porém em menor escala, inclusive com a geração de gráficos e alerta contra incêndios. Tais situações podem ser estendidas e reproduzidas em escala maior e em um ambiente real com as devidas alterações e adaptações de acordo com a necessidade.

A criação do dispositivo microcontrolador responsável pelas medições do ambiente permitiu a coleta dos dados de forma rápida e satisfatória, contribuindo de forma positiva para proteção do ambiente no qual se instalou.

O banco de dados em conjunto com a plataforma *web* de monitoramento contribuíram para uma visualização dos valores de medições e alarmes de forma intuitiva, direta e remota, facilitando o monitoramento dos dados e na compreensão visual de como o ambiente físico do *Data Center* se encontra. Além de contribuir com a geração de dados históricos que permita realizar análises estatísticas futuras.

Os testes feitos dentro do modelo representativo do *Data Center* foram um sucesso, ambos testes de temperatura, umidade e gases foram capazes de gerar efeitos significativos no ambiente físico, o que proporcionou validar o uso da central de monitoramento para a identificação de anomalias e problemas climáticos através de sua *interface web*.

Os resultados obtidos através dos testes se mostraram bastante satisfatórios, com tempos de respostas rápidos e com uma *interface* simples e intuitiva a central de monitoramento se mostrou bastante útil e eficaz para a identificação e análise das principais mudanças no ambiente físico de um *Data Center*, desempenhando assim sua função com destreza.

5.1 Sugestões de Trabalhos Futuros

Como sugestão de trabalhos futuros é recomendado que ocorra uma expansão e melhoria do sistema já criado. Por isso, são sugeridos os seguintes trabalhos:

1. Implementação de uma central de monitoramento para outros ambientes, onde seja necessário e relevante a análise do ambiente físico. Aumentando também a qualidade e quantidade dos sensores utilizados para realizar as medições do ambiente físico. Promovendo a geração de mais variáveis para a análise de comportamento do ambiente;
2. Criação de uma plataforma *web* de monitoramento de *Data Center* mais aprimorada, com diversos tipos de dados, análises e estatísticas a respeito dos valores medidos dentro do ambiente simulado ou real;
3. Aprimoramento dos alertas gerados tanto no modelo representativo do *Data Center*, como na plataforma web de monitoramento, com sinais sonoros e luminosos em ambos os locais. Além da imposição de limites bem definidos para a geração de alertas de temperatura, umidade e gases.

BIBLIOGRAFIA

ALGAR. Gráficos essenciais para o monitoramento de rede corporativa. **Algar Telecom**, 2017. Disponivel em: <https://blog.algartelecom.com.br/tecnologia/graficos-essenciais-para-o-monitoramento-de-rede-corporativa/>. Acesso em: 16 Outubro 2020.

APACHE. *Apache HTTP Server*. ***The Apache Software Foundation***, 2020. Disponivel em: <https://projects.apache.org/project.html?httpd-http_server>. Acesso em: 15 Outubro 2020.

ARDUINO. *Delay()*: Documentação de Referência do Arduino. **ARDUINO**, 2020. Disponivel em: <https://www.arduino.cc/reference/pt/language/functions/time/delay/>. Acesso em: 10 Outubro 2020.

ARDUINO. *Serial.begin()*: Documentação de Referência do Arduino. **ARDUINO**, 2020. Disponivel em: <https://www.arduino.cc/reference/pt/language/functions/communication/serial/begin/>. Acesso em: 10 Outubro 2020.

BOOTSTRAP. *Introduction.* ***GetBootstrap***, 2020. Disponivel em: <https://getbootstrap.com/docs/4.5/getting-started/introduction/>. Acesso em: 16 Outubro 2020.

CISCO. *What Is a Data Center.* **Cisco**, 2020. Disponivel em: <https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html>. Acesso em: 27 Setembro 2020.

COWAN, C.; GASKINS, C. *Monitoring Physical Threats in the Data Center.* **Schneider Electric**, 2011. Disponivel em: <https://download.schneider-electric.com/files?p_Doc_Ref=SPD_JMON-5ZLP8M_EN>. Acesso em: 24 Setembro 2020.

DOBRAN, B. *Data Center Security: Physical and Digital Layers of Protection*. **Phoenixnap**, 2019. Disponivel em: <https://phoenixnap.com/blog/data-center-security>. Acesso em: 04 Outubro 2020.

DRIEMEYER, R. **Projeto de melhoria de *Data Center* com ênfase em infraestrutura e eficiência energética**. UNIVATES. Lajeado, p. 131. 2016.

ELMASRI, R.; NAVATHE, S. **Sistema de banco de dados**. Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Addison Wesley, v. 1, 2011.

ELMASRI, R.; NAVATHE, S. B. **Sistema de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, v. 1, 2011.

ESTRELLA, C. O Que é PHP? Guia Básico de Programação PHP. **Hostinger**, 2020. Disponivel em: <https://www.hostinger.com.br/tutoriais/o-que-e-php-guia-basico/>. Acesso em: 16 Outubro 2020.

EVEO. Guia completo sobre *Data Center* para empresas. **Eveo**, 2019. Disponivel em: <https://www.eveo.com.br/blog/guia-completo-data-center/>. Acesso em: 28 Setembro 2020.

FARIAS, J. D. P. AJAX Tutorial. **Devmidia**, 2012. Disponivel em: <https://www.devmedia.com.br/ajax-tutorial/24797>. Acesso em: 15 Outubro 2020.

FELTER, B. *What Are the Most Important Data Center Security Standards.* **Vxchnge**, 2019. Disponível em: <https://www.vxchnge.com/blog/data-center-physical-security-standards>. Acesso em: 23 Setembro 2020.

FILHO, M. F. **Gestão da Infraestrutura do Datacenter: Livro digital**. UnisulVirtual. Palhoça, p. 65. 2016.

GARBIN, W. C. Identificar requisição POST ou GET com PHP. **Receitas de Código**, 2017. Disponivel em: <https://receitasdecodigo.com.br/php/identificar-requisicao-post-ou-get-com-php>. Acesso em: 02 Outubro 2020.

GONÇALVES, A. O que é CSS? Guia Básico para Iniciantes. **Hostinger**, 2019. Disponivel em: <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>. Acesso em: 16 Outubro 2020.

GOODRICH, M.; TAMASSIA, R. **Introdução a seguraça de redes**. 1. ed. Sçao Paulo: Bookman, v. 1, 2013.

GREENGARD, S. *10 Top Data Center Security Solutions.* **Datamation**, 2019. Disponivel em: <https://www.datamation.com/data-center/top-data-center-security-solutions.html>. Acesso em: 28 Setembro 2020.

GRIFFITH, E. *What Is Cloud Computing?* **Pcmag**, 2020. Disponivel em: <https://www.pcmag.com/news/what-is-cloud-computing>. Acesso em: 03 Outubro 2020.

GUIMARÃES, F. Enviar e receber dados pelo *Web* Server do NodeMcu. **Mundo Projetado**, 2018. Disponivel em: <http://mundoprojetado.com.br/enviar-e-receber-dados-pelo-*web*-server-do-nodemcu-aula-7-nb/>. Acesso em: 02 Outubro 2020.

HANÁK. *How Data Center Free Cooling Works and Why it is Brilliant.* **Masterdc**, 2015. Disponivel em: <https://www.masterdc.com/blog/what-is-data-center-free-cooling-how-does-it-work/>. Acesso em: 27 Setembro 2020.

ISRAEL. Como enviar valores para um banco de dados com o NODEMCU - ESP8266. PARTE 1. **Mecatrônica com Reciclagem**, 2019. Disponivel em: <https://youtu.be/9sk_qoJpD2o>. Acesso em: 03 Outubro 2020.

ISRAEL. Como enviar valores para um banco de dados com o NODEMCU - ESP8266. PARTE 2 (Final). **Mecatrônica com Reciclagem**, 2019. Disponivel em: <https://youtu.be/9sk_qoJpD2o>. Acesso em: 03 Outubro 2020.

JOHNSON, B. *How Data Centers Work*. **Howstuffworks**, 2020. Disponivel em: <https://computer.howstuffworks.com/data-centers1.htm>. Acesso em: 27 Setembro 2020.

LACROIX, L. Iniciando com Chart.js. **MundoJS**, 2018. Disponivel em: <https://www.mundojs.com.br/2018/01/18/iniciando-com-chart-js>. Acesso em: 16 Outubro 2020.

LONGEN, A. S. O que é Apache? Uma Visão Aprofundada do Servidor Apache. **Hostinger**, 2020. Disponivel em: <https://www.hostinger.com.br/tutoriais/o-que-e-apache>. Acesso em: 15 Outubro 2020.

MADEIRA, D. Proteus: Manual para simulação do Arduino. **Embarcados**, 2015. Disponivel em: <https://www.embarcados.com.br/proteus/>. Acesso em: 12 Outubro 2020.

MARIN, P. S. ***Data Centers* - desvendando cada passo:** projeto, infraestrutura física e eficiência energética. 1. ed. São Paulo: Érica, v. 1, 2013.

MARQUES, R. O que é HTML? Entenda de forma descomplicada. **Homehost**, 2019. Disponivel em: <https://www.homehost.com.br/blog/tutoriais/o-que-e-html/>. Acesso em: 01 Outubro 2020.

MIYADAIRA, A. N. **Microcontroladores PIC18:** Aprenda e Programe em Linguagem C. 4. ed. São Paulo: Érica, v. 1, 2009.

MONK, S. **Programação com Arduino:** Parte 2. 1. ed. Porto Alegre: Bookman, v. 2, 2015.

MONK, S. **Programação com Arduinos:** começando com sketches. 2. ed. Porto Alegre: Bookman, v. 1, 2017.

MURTA, G. Guia completo do NodeMCU: ESP12 Usando Arduino IDE (2). **Eletrogate**, 2018. Disponivel em: <https://blog.eletrogate.com/nodemcu-esp12-usando-arduino-ide-2/>. Acesso em: 12 Outubro 2020.

NASCIMENTO, B. D. S. MySQL Tutorial: Introdução ao Banco de Dados Relacional MySQL. **Devmedia**, 2015. Disponivel em: <https://www.devmedia.com.br/mysql-tutorial/33309>. Acesso em: 02 Outubro 2020.

OLIVEIRA, E. Como usar com Arduino - Sensor (Detector) de Gás Inflamável/Fumaça - MQ-2. **MasterWalker**, 2018. Disponivel em: <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-detector-de-gas-inflamavel-fumaca-mq-2/>. Acesso em: 23 Setembro 2020.

OLIVEIRA, G. NodeMCU - Uma plataforma com características singulares para o seu projeto ioT. **MasterWalker**, 2017. Disponivel em: <https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas-singulares-para-o-seu-projeto-iot/>. Acesso em: 23 Setembro 2020.

PALOALTO. *What is a Data Center?* **Paloalto Networks**, 2020. Disponivel em: <https://www.paloaltonetworks.com/cyberpedia/what-is-a-data-center>. Acesso em: 25 Setembro 2020.

PHP. *mysql_connect.* **PHP**, 2020. Disponivel em: <https://www.php.net/manual/pt_BR/function.mysql-connect.php>. Acesso em: 15 Outubro 2020.

PHP. O que o PHP pode fazer? **PHP**, 2020. Disponivel em: <https://www.php.net/manual/pt_BR/intro-whatcando.php>. Acesso em: 03 Outubro 2020.

PHPMYADMIN. *Bringing MySQL to the web*. **PHPMYADMIN**, 2020. Disponivel em: <https://www.phpmyadmin.net/>. Acesso em: 01 Outubro 2020.

POZZEBOM, R. O que foi a Bolha da *Internet*? **Oficina da Net**, 2015. Disponivel em: <https://www.oficinadanet.com.br/post/14314-o-que-foi-a-bolha-da-*Internet* >. Acesso em: 27 Setembro 2020.

REIS, T. Bolha da *Internet*: entenda as causas e consequências desse evento. **Suno**, 2019. Disponivel em: <https://www.sunoresearch.com.br/artigos/bolha-da-*Internet* />. Acesso em: 27 Setembro 2020.

RIBEIRO, W. R. **Asterisk Com Café**. 2. ed. São Paulo: Safecreative, v. 1, 2013.

ROSA, A. L. *Let's talk about Data Center Monitoring*. **Pandorafms**, 2018. Disponivel em: <https://pandorafms.com/blog/data-center-monitoring/>. Acesso em: 25 Setembro 2020.

ROSSI, F. *Data Center* I: Sistema Ininterrupto de Energia Elétrica. **Teleco**, 2013. Disponivel em: <https://www.teleco.com.br/tutoriais/tutorialdcseg1/pagina_6.asp>. Acesso em: 29 Setembro 2020.

SANT, J. A.; DAMASCENO, J. Curso completo de Desenvolvimento *Web* - Crie 6 projetos. **Udemy**, 2017. Disponivel em: <https://www.udemy.com/course/curso-completo-do-desenvolvedor-*web*/>. Acesso em: 15 Outubro 2020.

SILVEIRA, A. L. O Que É MySQL? Guia Para Iniciantes. **Hostinger**, 2019. Disponivel em: <https://www.hostinger.com.br/tutoriais/o-que-e-mysql/>. Acesso em: 03 Outubro 2020.

SOUZA, I. D. Entenda o que é HTTP e o quão importante esse protocolo é para o seu site. **Rockcontent**, 2019. Disponivel em: <https://rockcontent.com/br/blog/http/>. Acesso em: 02 Outubro 2020.

SOUZA, I. D. Entenda o que é jQuery, para que serve e como usar no seu site. **Rockcontent**, 2020. Disponivel em: <https://rockcontent.com/br/blog/jquery/>. Acesso em: 16 Outubro 2020.

SOUZA, I. D. phpMyAdmin: saiba o que é e aprenda como instalar e criar um banco de dados nele. **Rockcontent**, 2020. Disponivel em: <https://rockcontent.com/br/blog/phpmyadmin/>. Acesso em: 01 Outubro 2020.

STALLINGS, W.; BROWN, L. **Segurança de Computadores:** Princípios e Práticas. 2. ed. Rio de Janeiro: Elsevier Editora Ltda, v. 1, 2014.

SUMARES, G. *Data Centers*: tudo que você precisa saber. **Olhar Digital**, 2016. Disponivel em: <https://olhardigital.com.br/pro/noticia/data-centers-tudo-que-voce-precisa-saber/58506>. Acesso em: 25 Setembro 2020.

TAVARES, L. O que é um Servidor *Web* (*Web* Server). **Melhores Hospedagem de *Sites***, 2019. Disponivel em: <https://www.melhoreshospedagemde*sites*.com/servidor-*web*/>. Acesso em: 15 Outubro 2020.

THOMSEN, A. Monitorando Temperatura e Umidade com o sensor DHT11. **Filipeflop**, 2013. Disponivel em: <https://www.filipeflop.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/>. Acesso em: 23 Setembro 2020.

THOMSEN, A. Como programar o módulo ESP8266 NodeMCU. **Filipeflop**, 2016. Disponivel em: <https://www.filipeflop.com/blog/esp8266-nodemcu-como-programar/>. Acesso em: 27 Setembro 2020.

THOMSEN, A. Como programar o NodeMCU com IDE Arduino. **Filipeflop**, 2016. Disponivel em: <https://www.filipeflop.com/blog/programar-nodemcu-com-ide-arduino/>. Acesso em: 27 Setembro 2020.

VIDAL, V. Programação com Arduino: Parte 1. **Eletrogate**, 2017. Disponivel em: <https://blog.eletrogate.com/programacao-arduino-parte-1/>. Acesso em: 02 Outubro 2020.

WEBSLESSON. *Create Dynamic Column Chart using PHP Ajax with Google Charts*. **Webslesson**, 2018. Disponivel em: <https://www.*web*slesson.info/2018/07/create-dynamic-column-chart-using-php-ajax-with-google-charts.html>. Acesso em: 18 Outubro 2020.

XAMPP. XAMPP *Installers and Downloads for Apache Friends.* **Apache Friends**, 2020. Disponivel em: <https://www.apachefriends.org/pt_br/index.html>. Acesso em: 03 Outubro 2020.