UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL BENTO VILLANTI

LUCAS SASKOSKI

AUTOMAÇÃO E SEGURANÇA: UM SISTEMA DE MONITORAMENTO PARA DATA CENTERS

CURITIBA

2024

GABRIEL BENTO VILLANTI
LUCAS SASKOSKI

AUTOMAÇÃO E SEGURANÇA: UM SISTEMA DE MONITORAMENTO PARA DATA CENTERS

Automation and Security: A Monitoring System for Data Centers

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Orientadora: Prof.ª Dra. Luciane Agnoletti Dos Santos Pedotti

CURITIBA
2024

GABRIEL BENTO VILLANTI
LUCAS SASKOSKI

AUTOMAÇÃO E SEGURANÇA: UM SISTEMA DE MONITORAMENTO PARA DATA CENTERS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 25/março/2024

Luciane Agnoletti Dos Santos Pedotti
Prof.ª Dra.
Universidade Tecnológica Federal do Paraná

Clayton de Moura Belo
Prof. Mestre
Universidade Tecnológica Federal do Paraná

Omero Francisco Bertol
Prof. Mestre
Universidade Tecnológica Federal do Paraná

Rubens Alexandre de Faria
Prof. Dr.
Universidade Tecnológica Federal do Paraná

CURITIBA
2024

Esta conquista é dedicada às nossas famílias,
que sempre nos ofereceram apoio e incentivo
nos momentos mais desafiadores.

AGRADECIMENTOS

À Geraldo e Irene, meus pais incansáveis, aos meus irmãos, e à Lívia Vanessa, minha companheira de todas as horas, pela paciência e incentivo. Este trabalho é dedicado a vocês com toda minha gratidão. Dedico também a todos amigos conquistados nessa jornada, com um agradecimento especial aos amigos e professores da UTFPR do câmpus de Campo Mourão.

Lucas Saskoski

Gostaria de expressar minha mais profunda gratidão a todas as pessoas que estiveram ao meu lado durante esta jornada acadêmica, tornando possível a conclusão desta etapa.

Primeiramente, quero agradecer aos meus professores, cuja dedicação e orientação foram fundamentais para o meu crescimento intelectual. Em especial, quero expressar minha sincera gratidão à minha professora orientadora Dra. Luciane Agnoletti Dos Santos Pedotti, cujo apoio e orientação foram essenciais para o desenvolvimento deste trabalho e ao professor Omero Francisco Bertol, que me orientou durante os estágios e sempre esteve presente para conselhos e ajudar na solução de problemas.

Agradeço também à minha família, aos meus pais Marcelo e Silvana e à minha irmã por seu apoio incansável, amor incondicional e por sempre acreditarem em mim, mesmo nos momentos mais desafiadores. Não posso deixar de mencionar meus amigos Dhouglas, Vinicius, Ronaldo e Rafael, que estiveram ao meu lado durante toda essa jornada. Nos bons e nos maus momentos, vocês foram essenciais para que fosse possível alcançar meus objetivos.

Por fim, agradeço a todos que, de alguma forma, contribuíram para o meu crescimento acadêmico, pessoal e profissional. Sem dúvidas, foram anos de muito aprendizado, e estou profundamente grato por cada experiência vivida dentro da UTFPR.

Gabriel Bento Villanti

"Os obstáculos são aquelas coisas terríveis que você vê
quando desvia os olhos do seu objetivo."
Henry Ford

RESUMO

Este trabalho de conclusão de curso apresenta um sistema de monitoramento de data center que utiliza uma variedade de sensores e atuadores para garantir a eficiência operacional e a segurança do ambiente. Por meio da coleta contínua de dados sobre parâmetros críticos, como temperatura, gases, presença e umidade, os sensores fornecem informações valiosas para a tomada de decisões em tempo real. Complementarmente, os atuadores permitem a execução de ações automáticas para responder às condições detectadas, garantindo a integridade e o funcionamento adequado dos recursos do *data center*. Durante o projeto será abordado o desenvolvimento e implementação do sistema, destacando os desafios enfrentados e as estratégias adotadas para assegurar um ambiente confiável e eficiente.

Palavras-chave: *data center*; automatização; monitoramento; segurança.

ABSTRACT

This undergraduate thesis presents a data center monitoring system that utilizes a variety of sensors and actuators to ensure operational efficiency and environment safety. Through continuous data collection on critical parameters such as temperature, gases, presence, and humidity, the sensors provide valuable information for real-time decision-making. Additionally, the actuators enable the execution of automatic actions to respond to detected conditions, ensuring the integrity and proper functioning of the data center resources. Throughout the project, the development and implementation of the system will be addressed, highlighting the challenges faced and the strategies adopted to ensure a reliable and efficient environment.

Keywords: data center; automation; monitoring; security.

LISTA DE FIGURAS

Figura 1 – Sensor DHT22 23
Figura 2 – Sensor MQ-2 24
Figura 3 – Sensor HC SR501 25
Figura 4 – Sensor Reed 26
Figura 5 – Módulo Relê 4 Canais 27
Figura 6 – Buzzer 28
Figura 7 – Diagrama de Blocos das Conexões do Sistema de Monitoramento . . . 34
Figura 8 – Esquemático do MCP 36
Figura 9 – Diagrama de funcionamento 37
Figura 10 – Esquemático sensor de queda de energia 39
Figura 11 – Simulação sensor de queda de energia 40
Figura 12 – Arquitura do código - Pycharm 42
Figura 13 – Drivers - Pycharm 42
Figura 14 – Repositório no GitHub 46
Figura 15 – Repositório de Testes 47
Figura 16 – Desenho da PCI para o sensor de queda de energia 49
Figura 17 – Modelo 3D para o sensor falta de fase 50
Figura 18 – Desenho da PCI para o módulo Wi-Fi 51
Figura 19 – Modelo 3D para o módulo Wi-Fi 51
Figura 20 – Print da tela do aplicativo Trello 58
Figura 21 – Esquemático do módulo Wi-Fi 64

LISTA DE FOTOGRAFIAS

Fotografia 1 – Sensor de queda de energia montado **50**
Fotografia 2 – PCI corroida módulo Wi-Fi **52**
Fotografia 3 – Sensor módulo Wi-Fi **52**
Fotografia 4 – Protótipo montado **55**
Fotografia 5 – Vista frontal do protótipo **56**

LISTA DE TABELAS

Tabela 1 – Custos do protótipo **57**

LISTAGEM DE CÓDIGOS FONTE

Listagem 1 – Código Driver I2C **43**
Listagem 2 – Exemplo de código de teste de integração **48**
Listagem 3 – Classe TelegramBOT **54**
Listagem 4 – Arquivo de configuração: pinos MCP23017 . . . . . **66**

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CI	Circuito Integrado
CLI	Interface de linha de comando, do inglês *Command Line Interface*
CPU	Unidade Central de Processamento, do inglês *Central Processing Unit*
GPIO	Entrada/Saída de Propósito Geral, do inglês *General-Purpose Input/Output*
I/O	Entrada/Saída, do inglês *Input/Output*
I2C	Circuito Inter-integrado, do inglês *Inter-Integrated Circuit*
IoT	Internet das Coisas, do inglês *Internet of Things*
OOP	Programação orientada à objetos, do inglês *object-oriented programming*
PCB	Placa de Circuito Impresso, do inglês *Printed Circuit Board*
PCI	Placa de Circuito Impresso
PIR	Sensor de Infravermelho Passivo, do inglês *Passive Infrared Sensor*
RPi	Raspberry Pi
SPI	Interface Periférica Serial, do inglês *Serial Peripheral Interface*
UART	Transmissor/Receptor Assíncrono Universal, do inglês *Universal Asynchronous Receiver/Transmitter*
Wi-Fi	Rede sem Fio, do inglês *Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO 16

1.1 Objetivos 16

1.1.1 Objetivo geral 16

1.1.2 Objetivos específicos 17

1.2 Justificativa 17

1.3 Impactos Esperados 18

1.3.1 Tecnológicos 18

1.3.2 Científicos 19

1.3.3 Econômicos 19

1.3.4 Sociais 19

1.4 Estrutura do trabalho 20

2 REFERENCIAL TEÓRICO 21

2.1 Protocolos de Comunicação 21

2.1.1 Protocolo I2C 21

2.1.2 Protocolo Wi-Fi 22

2.2 Sensores 23

2.2.1 Sensor de Umidade e Temperatura: DHT22 23

2.2.2 Sensor de Gás e Fumaça: MQ-2 24

2.2.3 Sensor de Presença: HC SR501 24

2.2.4 Sensor de porta: Reed 25

2.2.5 Sensor de queda de energia 26

2.3 Atuadores 27

2.3.1 Módulo de relés 27

2.3.2 Dispositivos Sonoros de Emergência: Buzzer e/ou Sirene 27

3 MATERIAIS E METODOLOGIA 29

3.1 Materiais 29

3.1.1 Plataformas de Desenvolvimento 29

3.1.1.1 Microcontrolador: Raspberry pi 3 29

3.1.1.2 Expansor de GPIOs: MCP23017 30

3.1.1.3 Ambiente de desenvolvimento (IDE) 30

3.1.1.4 Ambiente de desenvolvimento de Hardware . . . 31
3.2 Metodologia . . . 32
3.2.1 Estudo dos recursos disponíveis do Raspberry Pi 3 . . . 32
3.2.2 Modelamento do sistema de controle . . . 32
3.2.3 Realização de testes em um protótipo de baixa potência . . . 33
3.2.4 Desenvolvimento da interface com o usuário . . . 33
4 DESENVOLVIMENTO . . . 34
4.1 Arquitetura do sistema embarcado . . . 34
4.2 Descrição de Funcionamento . . . 34
4.3 Modelagem do sistema . . . 35
4.4 Hardware . . . 37
4.4.1 Projeto do sensor falta de fase . . . 38
4.4.1.1 Simulação do sensor . . . 39
4.4.2 Módulo Rede sem Fio, do inglês *Wireless Fidelity* (Wi-Fi) com ESP-8266 . . . 40
4.5 Firmware . . . 41
4.5.1 Python e bibliotecas . . . 41
4.5.2 Desenvolvimento . . . 41
4.5.2.1 Arquitetura do Código . . . 41
4.5.2.2 Drivers . . . 42
4.5.2.3 Classe MCPGpio . . . 44
4.5.2.4 *Command Line Interface* . . . 44
4.5.2.5 Adaptador de arquivos - JSON . . . 44
4.5.2.6 Classes dos Sensores . . . 44
4.5.2.7 Serviços . . . 45
4.5.2.8 Controle de Versionamento . . . 46
4.5.3 Testes e validação . . . 46
5 RESULTADOS EXPERIMENTAIS . . . 49
5.1 Protótipo de Hardware . . . 49
5.2 Interface com usuário . . . 52
5.2.1 Página HTML . . . 52
5.2.2 Robô mensageiro: BOT Telegram . . . 53
5.3 Montagem do protótipo . . . 55

6 GESTÃO DE PROJETO 58
6.1 Gerenciamento de tarefas 58
6.2 Melhorias do Projeto 59
7 CONCLUSÃO 60
REFERÊNCIAS 61
APÊNDICE A ESQUEMÁTICO DO MÓDULO WI-FI 64
APÊNDICE B ARQUIVO DE CONFIGURAÇÃO JSON 66

1 INTRODUÇÃO

O avanço da internet, bem como o aumento frequente do número de pessoas e dispositivos conectados a ela, trouxeram uma crescente necessidade de armazenamento e processamento de dados. Nesse contexto, projetos tecnológicos são constantemente desenvolvidos para atender a essa demanda. Os elementos de rede responsáveis por concentrar e fornecer capilaridade a essa operação estão localizados em um ambiente conhecido como data center.

Percebe-se a criticidade e a importância de manter a saúde operacional desses centros, assim como dos elementos que os compõem. Portanto, é imprescindível monitorar o maior número possível de variáveis ambientais, a fim de evitar interferências ou falhas nesse ambiente. Por exemplo, a temperatura deve ser monitorada e controlada para evitar o superaquecimento dos dispositivos eletrônicos. Além disso, a umidade deve ser mantida em níveis ideais para prevenir a oxidação dos metais presentes no ambiente, especialmente dos conectores e cabos.

Já a segurança do controle de acesso também é crucial, considerando que os dados que circulam no datacenter podem ser sensíveis, e proteger os equipamentos contra acesso não autorizado é fundamental para evitar prejuízos financeiros. Diante deste cenário, este trabalho tem como objetivo desenvolver um sistema de monitoramento robusto e expansível para parâmetros de umidade, temperatura e segurança de acesso em data centers. Esse sistema fará uso de sensoriamento e um aplicativo supervisório, visando garantir o bom funcionamento e a integridade desse ambiente crítico em tempo real.

1.1 Objetivos

Nesta seção, apresenta-se os objetivos fundamentais deste trabalho, centrados na investigação das interseções entre automação e segurança, explorando como o conceito de automação pode ser implementado de forma segura em diferentes contextos.

1.1.1 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema abrangente de monitoramento para *data centers*, visando garantir a segurança, a eficiência energética e a disponibilidade dos recursos físicos. Especificamente, pretende-se utilizar sensores de temperatura, umidade, gás, fumaça e controle de acesso físico por meio de sensor de presença e sensor de abertura de porta, além de sensor de corrente para monitorar o ar condicionado e de falta de fase verificar queda de energia. Para alcançar esse objetivo, será utilizado um microcontrolador, juntamente com o expansor de GPIO, para controlar e interagir com os diversos sensores e atuadores do sistema. Serão desenvolvidos dispositivos de *hardware* para a acomodação adequada dos leitores e um *firmware* para a leitura e o controle desses dispositivos. A leitura de temperatura e

umidade será feita utilizando módulos com comunicação *Wi-fi* de forma a posicionar os sensores de acordo com *layout* do espaço físico. Já os sensores de porta presença e fumaça, bem como os alarmes e iluminação, serão cabeados e gerenciados pelo expansor de GPIO. Além disso, será implementado um sistema de interface com o usuário que permitirá a visualização e o gerenciamento dos dados coletados, facilitando o monitoramento e as tomadas de decisão pelos responsáveis pelo *data center*.

1.1.2 Objetivos específicos

- Compreender quais as necessidades de operação de um *data center*;
- Estabelecer medidas de temperatura e umidade;
- Determinar medidas de segurança para controle de acesso físico;
- Desenvolver *hardware* embarcado para acomodação de sensores e do controlador;
- Implementar *firmware* para leitura e controle dos sensores;
- Implementar *firmware* para acionamento dos atuadores;
- Integrar um ambiente supervisório para o sistema;
- Criar um sistema autônomo de monitoramento;
- Estabelecer um canal de comunicação em tempo real;
- Produzir uma interface de usuário.

1.2 Justificativa

O desenvolvimento deste projeto de monitoramento de *data centers* foi motivado pelas necessidades e desafios enfrentados pelas organizações na gestão eficiente e segura de suas infraestruturas tecnológicas. O monitoramento e a gestão desses ambientes são fundamentais para garantir a economia, a disponibilidade e a segurança das operações, como destacado por Mauro Filho (FILHO, 2016).

A importância do projeto para a sociedade está na proteção das informações e dos ativos das organizações. Conforme mencionado por Sêmola (SÊMOLA, 2003), as ameaças podem ser classificadas em três camadas: físicas, tecnológicas e humanas. Os *data centers* estão expostos a riscos como incêndios, inundações, invasões virtuais e ações humanas físicas. Essas ameaças podem comprometer a confidencialidade, integridade e disponibilidade das informações, afetando os negócios das organizações, como destacado por Ferreira (FERREIRA, 2003).

Ao comparar o projeto proposto com as soluções existentes, destacam-se algumas vantagens significativas. Primeiramente, o sistema oferece a vantagem da personalização, permitindo que seja adaptado às necessidades específicas de cada organização. Isso significa que os parâmetros de monitoramento e alertas podem ser configurados de acordo com os requisitos e políticas de segurança internas de cada empresa. Outra vantagem é o monitoramento em tempo real, que é uma funcionalidade essencial para garantir a detecção precoce de problemas. O sistema é capaz de monitorar continuamente as condições do *data center*, identificando quaisquer anomalias ou incidentes assim que são deflagrados. Isso permite que sejam gerados alertas imediatos, possibilitando uma ação rápida e eficiente para mitigar os impactos negativos.

A linguagem Python é orientada a objetos (Programação orientada à objetos, do inglês *object-oriented programming* (OOP)) proporciona uma estrutura flexível e modular para o desenvolvimento deste projeto, permitindo a criação de classes, objetos e métodos que encapsulam a lógica do sistema e promovem a reutilização de código. Essa abordagem facilita a manutenção, extensibilidade e escalabilidade do sistema de monitoramento, tornando-o mais eficiente e adaptável às necessidades específicas de cada organização. Combinada com a compatibilidade com o sistema operacional Linux, o Python oferece um ambiente robusto e confiável para a execução do projeto, contribuindo para a proteção e gestão eficiente dos *data centers* (LAMBERT, 2022).

1.3 Impactos Esperados

1.3.1 Tecnológicos

O sistema de monitoramento em um *data center* é de grande importância tecnológica, pois permite a identificação precoce de problemas, melhora o planejamento e a escalabilidade, otimiza a eficiência operacional e fornece informações valiosas para a tomada de decisões estratégicas. Essas características contribuem para o funcionamento confiável, seguro e eficiente do *data center*, apoiando o sucesso das operações tecnológicas de uma organização.

Um sistema de monitoramento contínuo permite identificar e diagnosticar problemas e falhas no *data center* de forma rápida e eficiente. Isso inclui alertas de temperatura elevada, queda de energia, falhas de *hardware*, entre outros. Ao detectar problemas precocemente, o sistema em conjunto com a equipe de operações pode agir rapidamente para resolver as questões antes que elas se tornem maiores ou causem interrupções significativas nos serviços.

Pensando sobre a otimização da eficiência operacional, um sistema de monitoramento detalhado fornece dados sobre o uso de energia, refrigeração e outros recursos críticos. Com base nesses dados, é possível implementar estratégias de eficiência energética, como ajustes nas configurações de resfriamento e acesso de pessoal autorizado.

1.3.2 Científicos

Um sistema de monitoramento em um *data center* desempenha um papel crucial ao fornecer dados e informações valiosas para estudos e pesquisas relacionadas à infraestrutura tecnológica. A análise e otimização de eficiência pode ser feita, visto que, o monitoramento detalhado de um *data center* possibilita o estudo científico da eficiência operacional e energética. Os dados coletados podem ser usados para analisar o desempenho energético de diferentes componentes, identificar áreas de desperdício e desenvolver estratégias para otimizar o uso de recursos. Essas pesquisas científicas contribuem para o desenvolvimento de práticas e políticas de sustentabilidade, buscando reduzir o consumo de energia e minimizar o impacto ambiental dos *data centers*.

Já no ponto de vista da segurança e resiliência, a segurança é uma preocupação fundamental em *data centers*, e a pesquisa científica nessa área é crucial para identificar vulnerabilidades, desenvolver técnicas de detecção de ameaças e propor soluções de mitigação. O sistema de monitoramento fornece dados relevantes para a análise científica da segurança dos *data centers*, permitindo o estudo de ameaças cibernéticas, monitoramento de acesso físico e aprimoramento das práticas de segurança.

1.3.3 Econômicos

O projeto tem como objetivo trazer resultados econômicos significativos para as organizações. Espera-se que o sistema de monitoramento contribua para a redução de custos operacionais, prevenção de danos, melhoria da eficiência operacional, aumento da confiabilidade dos serviços e uma melhor gestão dos recursos. Com a capacidade de identificar e resolver proativamente problemas, como superaquecimento, falhas de energia, o projeto visa evitar danos aos equipamentos, reduzir gastos com energia elétrica e manutenção, minimizar o tempo de inatividade e aumentar a produtividade. Além disso, ao fornecer informações detalhadas sobre o consumo de energia, utilização de recursos e desempenho dos equipamentos, o sistema de monitoramento auxilia na tomada de decisões informadas para uma melhor gestão dos recursos.

Com esses benefícios, espera-se que o projeto proporcione resultados econômicos positivos para as organizações.

1.3.4 Sociais

Do ponto de vista social, o monitoramento de um *data center* desempenha um papel crucial na garantia da disponibilidade e confiabilidade dos serviços essenciais, na proteção da

segurança dos dados, na promoção da eficiência energética e no monitoramento do impacto ambiental.

Um *data center* é responsável por hospedar e fornecer diversos serviços essenciais, como armazenamento de dados e aplicativos. O monitoramento contínuo garante a disponibilidade e o bom funcionamento desses serviços, sendo vital para a produtividade de empresas, instituições governamentais e serviços críticos, como saúde e finanças. Além disso, o monitoramento protege os dados sensíveis, identificando ameaças de segurança e prevenindo prejuízos financeiros e danos à reputação. Também auxilia na eficiência energética, reduzindo custos e impactos ambientais, e no cumprimento de regulamentações ambientais.

1.4 Estrutura do trabalho

Este trabalho está organizado da seguinte forma:

- **Referencial Teórico:** O Capítulo 2 discute os fundamentos teóricos relacionados ao tema do projeto, apresentando conceitos, teorias e trabalhos anteriores relevantes.
- **Metodologia:** No Capítulo 3, é detalhada a metodologia utilizada na realização do trabalho, incluindo a abordagem de pesquisa, técnicas de coleta e análise de dados, entre outros aspectos.
- **Desenvolvimento:** O Capítulo 4 descreve o desenvolvimento do projeto em si, incluindo a implementação prática das ideias e conceitos discutidos anteriormente.
- **Resultados Experimentais:** No Capítulo 5, são apresentados os resultados obtidos a partir da implementação do projeto, incluindo análises, gráficos e interpretações dos dados coletados.
- **Gestão:** No Capítulo 6, é discutida a gestão do projeto, incluindo o planejamento, organização e execução das atividades, bem como a alocação de recursos e o controle do progresso.
- **Conclusões:** Finalmente, no Capítulo 7, são apresentadas as conclusões do trabalho, incluindo uma síntese dos resultados, discussão sobre as contribuições e limitações do estudo, e sugestões para futuras pesquisas.

2 REFERENCIAL TEÓRICO

2.1 Protocolos de Comunicação

Durante o projeto, foram explorados protocolos de comunicação essenciais, para facilitar a troca de dados entre os dispositivos. A escolha desses protocolos foi baseada nas necessidades do sistema e na compatibilidade com os dispositivos utilizados.

2.1.1 Protocolo I2C

A comunicação entre dispositivos é fundamental em muitas aplicações, especialmente em sistemas embarcados, onde recursos limitados e restrições de espaço são comuns. Uma das interfaces de comunicação mais utilizadas nesse contexto é o *Inter-Integrated Circuit* (I2C).

O protocolo Circuito Inter-integrado, do inglês *Inter-Integrated Circuit* (I2C), desenvolvido pela Philips Semiconductors (agora NXP Semiconductors), é uma interface serial de comunicação síncrona, que permite a conexão de vários dispositivos a um microcontrolador ou microprocessador usando apenas duas linhas de comunicação: uma para dados (*SDA - Serial Data Line*) e outra para o relógio (*SCL - Serial Clock Line*). Isso resulta em uma comunicação eficiente e simplificada em comparação com outras interfaces mais complexas, como SPI (*Serial Peripheral Interface*) ou UART (*Universal Asynchronous Receiver-Transmitter*) como foi dito por (VALDEZ; BECKER, 2015).

O MCP23017 é um expansor de portas GPIO (*General Purpose Input/Output*) que utiliza o protocolo I2C para comunicação com um microcontrolador ou outro dispositivo mestre. Para conectar um dispositivo MCP23017 a um controlador mestre usando o protocolo I2C, são necessárias algumas etapas:

- Configurar o endereço I2C:
    1. Identificar o endereço I2C do dispositivo que se deseja comunicar.
    2. Consultar o *datasheet* do dispositivo para obter o endereço I2C.
    3. Utilizar comandos ou bibliotecas específicas da linguagem de programação para configurar o endereço do dispositivo.

- Configurar o barramento I2C:

    1. Verificar a disponibilidade de portas I2C no microcontrolador ou dispositivo mestre.
    2. Conectar os dispositivos ao barramento I2C de acordo com a topologia desejada (por exemplo, em série ou em paralelo).
    3. Utilizar resistores de pull-up adequados para garantir a integridade do sinal no barramento I2C.
    4. Inicializar e configurar o barramento I2C no código do programa, definindo a velocidade de comunicação e outros parâmetros quando necessários.

2.1.2 Protocolo Wi-Fi

O protocolo *Wi-Fi* é uma tecnologia de comunicação sem fio que se tornou muito utilizada em ambientes residenciais, comerciais e industriais. Sua popularidade decorre da sua capacidade de oferecer conectividade de rede rápida e confiável, permitindo a interconexão de uma variedade de dispositivos, desde smartphones e laptops até dispositivos IoT (Internet das Coisas) e equipamentos de automação residencial. O *Wi-Fi* opera em frequências de rádio na faixa de 2,4 GHz e/ou 5 GHz, e é projetado para suportar transferências de dados de alta velocidade, proporcionando aos usuários uma experiência de rede fluida e sem interrupções.

A arquitetura do protocolo *Wi-Fi* é baseada em uma série de padrões definidos pelo *Institute of Electrical and Electronics Engineers* (IEEE), comumente conhecidos como as especificações da família IEEE 802.11, como definido por (STALLINGS, 2013). Esses padrões estabelecem as diretrizes para a implementação de redes sem fio, incluindo aspectos como modulação, codificação, gerenciamento de acesso ao meio e segurança. O funcionamento de redes sem fio envolve a transmissão de dados em pacotes por meio de ondas de rádio, utilizando técnicas avançadas de modulação para maximizar a eficiência espectral e minimizar interferências.

O ESP8266 é um microcontrolador de baixo custo e baixo consumo de energia, projetado para conectar dispositivos à internet por meio de redes sem fio. Ele se destaca por sua versatilidade e facilidade de uso, tornando-se uma escolha popular para projetos de IoT (Internet das Coisas) e automação residencial.

O funcionamento do ESP8266 é baseado em um processador de microcontrolador integrado com suporte embutido para conexão *Wi-Fi*. Este microcontrolador inclui uma unidade de processamento central (Unidade Central de Processamento, do inglês *Central Processing Unit* (CPU)), memória flash para armazenamento de programas e dados, memória RAM para armazenamento temporário de dados e periféricos de Entrada/Saída, do inglês *Input/Output* (I/O) para comunicação com outros dispositivos.

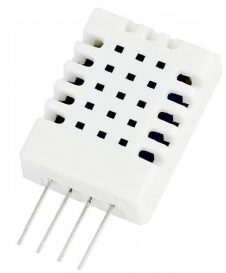
2.2 Sensores

Sensores desempenham um papel fundamental em projetos que envolvem a Internet das Coisas (Internet das Coisas, do inglês *Internet of Things* (IoT)), uma vez que são os olhos e ouvidos dos dispositivos conectados à rede. Esses dispositivos capturam dados do ambiente, como temperatura, umidade, luz, movimento e muito mais (MASCHIETTO *et al.*, 2021). Esses dados são então transmitidos através do sistema para análise e tomada de decisões em tempo real que possibilitam a automação, monitoramento e controle remoto de dispositivos, melhorando a eficiência, economizando recursos e tornando os sistemas IoT mais inteligentes e responsivos.

2.2.1 Sensor de Umidade e Temperatura: DHT22

Quando se analisa o sensor DHT22, também conhecido como AM2302, é evidente que ele oferece diversas vantagens significativas em aplicações de medição de temperatura e umidade. Essas vantagens incluem maior exatidão, uma faixa de medição mais ampla, construção mais robusta e uma comunicação digital mais estável. Em termos de precisão, o DHT22 possui ±0,5℃ para temperatura e ±2% para umidade (AOSONG, s.d.). Quanto à faixa de medição, pode medir em uma faixa de temperatura mais ampla, geralmente de -40℃ a 80℃ (AOSONG, s.d.), tornando-o mais adequado para ambientes com temperaturas extremas. Além disso, sua construção utiliza componentes de maior qualidade, o que o torna mais durável e confiável em aplicações a longo prazo. Em relação à comunicação, o DHT22 utiliza um protocolo digital mais estável, resultando em menos erros de leitura e maior robustez em ambientes ruidosos. Em resumo, o sensor DHT22 é uma escolha superior em relação a outros sensores quando se busca precisão, faixa de medição ampla e confiabilidade em aplicações de medição de temperatura e umidade.

Figura 1 – Sensor DHT22



Fonte: Adaptado de Usainfo[1].

[1] Disponível em: <https://www.usinainfo.com.br/sensor-de-umidade-arduino/sensor-de-umidade-e-temperatura-am2302-dht22-40-a-80c-6256.html>. Acesso em: 09 mar. 2024.

2.2.2 Sensor de Gás e Fumaça: MQ-2

O sensor MQ-2 é um dispositivo amplamente utilizado para detecção de gases inflamáveis e fumaça em diversas aplicações, incluindo sistemas de segurança, detecção de vazamentos de gás e monitoramento ambiental (HANWEI, s.d.). Opera com base no princípio de detecção de gás por meio de resistência elétrica, contendo um elemento sensor que é aquecido eletricamente. Quando o gás alvo está presente no ambiente, o elemento sensor reage quimicamente com o gás, resultando em uma mudança na resistência elétrica do sensor. Esta mudança na resistência é então medida e interpretada como a presença e concentração do gás detectado.

Entre os gases que o sensor MQ-2 pode detectar estão o gás metano (CH4), o gás butano (C4H10), o gás de hidrogênio (H2) e outros gases inflamáveis em concentrações específicas. No entanto, é importante notar que o sensor MQ-2 pode ser sensível a outros gases e vapores, o que pode levar a falsos positivos ou falsos negativos em determinadas condições.

Figura 2 – Sensor MQ-2

Fonte: Adaptado de Cytron[2].

2.2.3 Sensor de Presença: HC SR501

O Sensor de Movimento HC-SR501 foi desenvolvido para detectar movimentos em uma área específica. Ele é comumente empregado em projetos de automação residencial, sistemas de segurança, iluminação automática e outras aplicações que exigem detecção de movimento (MASCHIETTO *et al.*, 2021).

O HC-SR501 é um Sensor de Infravermelho Passivo, do inglês *Passive Infrared Sensor* (PIR), o que significa que ele detecta movimentos com base na variação do calor emitido pelos objetos em seu campo de visão. Ele consiste em um sensor PIR, que contém uma lente óptica e um elemento sensor que detecta mudanças na radiação infravermelha. Quando um movimento

[2] Disponível em: <https://my.cytron.io/c-sensor/p-mq2-smoke-lpg-co-sensor-module>. Acesso em: 09 mar. 2024.

é detectado, o sensor produz um sinal de saída que pode ser usado para acionar dispositivos externos, como luzes, alarmes ou sistemas de segurança.

Este sensor uma ampla faixa de detecção, geralmente entre 3 a 7 metros, com um ângulo de detecção de cerca de 120 graus, possuindo também ajustes de sensibilidade e tempo de retardo, o que permite personalizar sua operação de acordo com as necessidades específicas do projeto.

Figura 3 – Sensor HC SR501

Fonte: Adaptado de Eletrogate[3].

2.2.4 Sensor de porta: Reed

O sensor Reed, ou interruptor Reed, é um dispositivo elétrico usado para detectar a presença ou ausência de um campo magnético. Ele é constituído por duas lâminas de metal ferromagnético, normalmente feitas de níquel-ferro ou uma liga similar, que são separadas por um pequeno espaço e encapsuladas em um invólucro hermético.

Quando um campo magnético é aplicado ao sensor Reed, as lâminas metálicas são atraídas uma pela outra, fazendo com que se conectem e criem um circuito elétrico (FRANCHI, 2015). Esse circuito pode ser usado para ativar ou desativar dispositivos eletrônicos, como alarmes, sensores de posição, interruptores de segurança, entre outros .

Uma das principais vantagens dos sensores Reed é sua simplicidade e confiabilidade, o que significa que têm uma vida útil longa e são menos propensos a falhas devido ao desgaste. Além disso, sua operação é instantânea, proporcionando uma resposta rápida à presença do campo magnético.

[3] Disponível em: <https://www.eletrogate.com/sensor-de-movimento-presenca-pir>. Acesso em: 09 mar. 2024.

Figura 4 – Sensor Reed

Fonte: Adaptado de Ryndack[4].

2.2.5 Sensor de queda de energia

O sensor de falta de fase, também conhecido como sensor de falta de energia ou sensor de ausência de fase, é um dispositivo utilizado para detectar a interrupção ou falta de uma das fases em um sistema elétrico (MATTEDE, s.d.). Esse sensor desempenha um papel de prevenção de danos em equipamentos elétricos e na segurança dos sistemas de energia.

É projetado para monitorar continuamente o fornecimento de energia e detectar qualquer interrupção em uma das fases. Quando uma falta de fase é detectada, o sensor pode acionar um alarme audível ou visual, interromper o fornecimento de energia para os equipamentos conectados ou ativar um sistema de backup para evitar danos ou interrupções no funcionamento do sistema.

Existem diferentes tipos de sensores de falta de fase disponíveis, incluindo sensores baseados em relés, dispositivos eletrônicos de detecção de fase e sistemas de monitoramento remoto que utilizam tecnologias de comunicação sem fio. Alguns sensores também podem ser integrados a sistemas de automação industrial ou sistemas de gerenciamento de energia para fornecer funcionalidades avançadas, como registro de dados, diagnóstico de falhas e controle remoto.

A implementação de sensores de falta de fase é especialmente importante em aplicações críticas, como em indústrias, instalações comerciais, hospitais, *data centers* e outras infraestruturas que dependem de uma alimentação elétrica confiável. Além disso, esses sensores desempenham um papel fundamental na conformidade com os regulamentos de segurança elétrica e na prevenção de acidentes e danos materiais.

[4] Disponível em: <https://www.ryndackcomponentes.com.br/reed-switch-na-sensor-magnetico.html>. Acesso em: 09 mar. 2024.

2.3 Atuadores

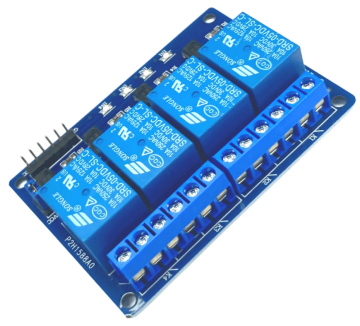
Os atuadores são componentes vitais em sistemas automatizados, convertendo sinais de controle em ações físicas. De motores a válvulas, eles executam tarefas específicas em resposta a comandos elétricos ou eletrônicos.

2.3.1 Módulo de relés

Os módulos de relés com 4 ou mais canais são amplamente utilizados em projetos de automação e controle, oferecendo uma solução versátil e confiável para a gestão de múltiplos dispositivos. Sua aplicabilidade abrange desde ambientes residenciais, onde podem ser utilizados para controlar luzes, eletrodomésticos e sistemas de segurança, até aplicações industriais, onde podem ser empregados para acionar motores, bombas e solenoides (FRANCHI, 2015).

A interface de controle simples presente nos módulos de relés facilita sua integração com sistemas microcontrolados, como Arduino, Raspberry Pi e outros dispositivos semelhantes. Isso permite que os usuários criem sistemas inteligentes e conectados à internet, onde os dispositivos controlados podem ser acionados e monitorados remotamente. Além disso, a presença de relés eletromagnéticos em cada canal oferece isolamento elétrico entre o circuito de controle e os dispositivos controlados, garantindo a segurança do sistema e protegendo os dispositivos de danos elétricos. A configuração e instalação desses módulos são geralmente simples, o que os torna uma escolha conveniente para aplicações de prototipagem e desenvolvimento rápido de sistemas.

Figura 5 – Módulo Relê 4 Canais



Fonte: Adaptado de a2robotics[5].

2.3.2 Dispositivos Sonoros de Emergência: Buzzer e/ou Sirene

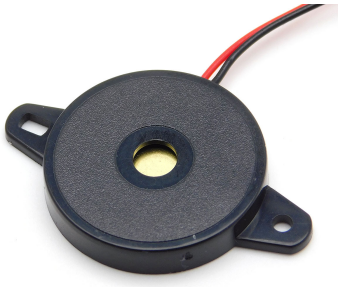
Um *buzzer* ou sirene de emergência é um dispositivo sonoro utilizado para alertar sobre situações críticas ou de emergência. Ele emite um som contínuo ou intermitente, geralmente de

[5] Disponível em: <https://www.a2robotics.com.br/modulo-rele-04-canais-5v>. Acesso em: 09 mar. 2024.

alta intensidade, para chamar a atenção das pessoas em uma determinada área e comunicar a necessidade de ação imediata.

Esses dispositivos são amplamente empregados em sistemas de segurança, alarmes de incêndio, ambulâncias, veículos de emergência e outras aplicações onde é crucial alertar rapidamente as pessoas sobre uma situação de perigo ou emergência (FRANCHI, 2015).

Figura 6 – Buzzer



Fonte: Adaptado de Nettigo[6].

[6] Disponível em: <https://nettigo.eu/products/piezoelectric-buzzer-with-leads>. Acesso em: 09 mar. 2024.

3 MATERIAIS E METODOLOGIA

A pesquisa exploratória com abordagem de pesquisa experimental proposta neste trabalho tem como objetivo desenvolver um sistema abrangente de monitoramento para *data centers*, visando garantir a segurança, a eficiência energética e a disponibilidade dos recursos físicos. Esse sistema utiliza sensores de temperatura, umidade, gás e controle de acesso físico para coletar dados relevantes em tempo real. Além disso, serão utilizados dispositivos Raspberry Pi 3 e um software para o controle e a visualização dos dados coletados.

3.1 Materiais

Nesta seção, serão apresentados os materiais e componentes utilizados no desenvolvimento do projeto proposto. Será destacado especialmente o microcontrolador Raspberry Pi, um dos elementos centrais da solução, e o expansor de GPIOs MCP23017, que desempenha um papel fundamental na expansão das capacidades de I/O do sistema. Além disso, serão discutidos outros periféricos e dispositivos empregados, fornecendo uma visão abrangente dos materiais e tecnologias utilizadas na implementação do sistema de automação e monitoramento proposto.

3.1.1 Plataformas de Desenvolvimento

3.1.1.1 Microcontrolador: Raspberry pi 3

A Raspberry Pi 3 desempenha um papel central como uma plataforma versátil e acessível para controle e processamento de dados. Equipada com uma variedade de interfaces de comunicação, incluindo Entrada/Saída de Propósito Geral, do inglês *General-Purpose Input/Output* (GPIO), I2C, Interface Periférica Serial, do inglês *Serial Peripheral Interface* (SPI) e Transmissor/Receptor Assíncrono Universal, do inglês *Universal Asynchronous Receiver/Transmitter* (UART), a Raspberry Pi 3 oferece uma ampla gama de opções de conectividade para interagir com dispositivos externos, como sensores e atuadores (FUTURE PUBLISHING LTD, 2016).

Especificamente, a comunicação I2C (Inter-Integrated Circuit) é uma das interfaces mais utilizadas para conectar dispositivos periféricos à Raspberry Pi 3. Com sua arquitetura simples de dois fios (SCL e SDA), o barramento I2C permite uma comunicação eficiente e confiável entre a Raspberry Pi e uma variedade de dispositivos.

A capacidade de se comunicar via I2C torna a Raspberry Pi 3 uma escolha ideal para projetos de automação e monitoramento, pois permite a fácil integração de uma ampla gama de dispositivos sensores e a coleta de dados em tempo real. Além disso, a Raspberry Pi 3 oferece

potência computacional suficiente para processar dados coletados pelos sensores, executar algoritmos de controle e tomar decisões em tempo real (FUTURE PUBLISHING LTD, 2016).

Portanto, a integração da Raspberry Pi 3 com a comunicação I2C oferece aos pesquisadores e desenvolvedores uma base sólida para a criação de soluções inovadoras em uma variedade de aplicações, desde a automação residencial até o monitoramento industrial.

3.1.1.2 Expansor de GPIOs: MCP23017

O MCP23017 (MICROCHIP, 2005) é um Circuito Integrado (CI) amplamente utilizado na eletrônica, especialmente em aplicações que envolvem a expansão de portas de entrada/saída em microcontroladores ou microprocessadores. Ele pertence à família de dispositivos da Microchip Technology e oferece uma solução eficiente para aumentar o número de pinos disponíveis para conectividade em sistemas embarcados, tornando a aplicação modular.

Este dispositivo é um expansor de I/O bidirecional de 16 bits que se comunica via barramento I2C, o que o torna compatível com uma ampla gama de plataformas microcontroladoras. As principais características do MCP23017 incluem:

- 16 Pinos I/O Bidirecionais: O MCP23017 possui 16 pinos de I/O que podem ser configurados individualmente como entradas ou saídas. Isso proporciona uma expansão significativa de I/O para sistemas com restrição de pinos.
- Comunicação I2C: O MCP23017 utiliza o protocolo I2C para comunicação com o microcontrolador ou microprocessador principal. Isso simplifica a conexão e o controle do dispositivo.
- Registradores Internos: O dispositivo possui registradores internos que permitem configurar a direção de cada pino (entrada ou saída), bem como o estado de cada pino individualmente. Isso oferece grande flexibilidade no controle dos pinos.
- Interrupções: O MCP23017 suporta interrupções que podem ser configuradas para notificar o microcontrolador quando ocorrerem mudanças em pinos específicos, economizando recursos de processamento e energia.
- Baixo Consumo de Energia: É projetado para ser eficiente em termos de energia, o que o torna adequado para aplicações alimentadas por bateria ou onde a economia de energia é crítica.

3.1.1.3 Ambiente de desenvolvimento (IDE)

O Ambiente de Desenvolvimento Integrado, ou IDE, é uma ferramenta de *software* que fornece um ambiente completo para desenvolvedores de software criarem, depurarem e gerenciarem seus projetos de programação. Essas ferramentas são projetadas para melhorar a

produtividade dos desenvolvedores, oferecendo uma série de recursos integrados, como um editor de código, depurador, gerenciador de projetos e ferramentas de compilação, tudo em um único pacote.

A principal IDE utilizada durante o projeto foi o PyCharm (JETBRAINS, 2019), justamente por fornecer análises de código, um depurador gráfico, um testador de unidade integrado, integração com sistemas de controle de versão, e suporta desenvolvimento web com Django. O PyCharm é uma ferramenta desenvolvida pela JetBrains, projetada especificamente para desenvolvedores Python, linguagem responsável por quase a totalidade do código fonte deste projeto. Ele é amplamente reconhecido por sua eficiência e rica variedade de recursos. Alguns aspectos notáveis que justificam a escolha desse editor de código avançado:

- Oferece um editor de código altamente personalizável, com recursos como realce de sintaxe, formatação automática, verificação de código, preenchimento automático e refatoração de código, tornando a escrita de código Python mais eficiente.
- Integração com Ambientes Virtuais: A IDE facilita a criação e gerenciamento de ambientes virtuais Python, que são úteis para isolar projetos e suas dependências. O foi essencial para integração com o sistema Linux, da RaspberryPi.
- Integração com Controle de Versão: A IDE oferece integração com sistemas de controle de versão populares, como Git, facilitando o controle de alterações em seu código.
- Suporte a Frameworks Python: PyCharm é compatível com uma variedade de frameworks Python, incluindo Django, Flask e Pyramid, oferecendo suporte à criação de aplicativos web Python de forma eficiente.
- Ferramentas de Teste Integradas: A IDE inclui suporte integrado para testes unitários, testes de integração e testes de aceitação, facilitando a escrita e execução de testes de qualidade de código.

De maneira geral, o PyCharm é uma IDE altamente especializada e robusta projetada para melhorar a produtividade dos desenvolvedores Python, tornando o desenvolvimento de software Python mais eficiente e agradável.

3.1.1.4 Ambiente de desenvolvimento de Hardware

No contexto do desenvolvimento de sistemas de *hardware*, o uso de *softwares* especializados são ferramentas essenciais para diversas etapas do processo. O Autodesk Eagle é uma solução amplamente reconhecida para o *design* de esquemáticos e placas de circuito impresso Placa de Circuito Impresso (PCI). Com sua interface intuitiva e recursos avançados, como a capacidade de criar símbolos personalizados e gerar layouts de Placa de Circuito Impresso, do inglês *Printed Circuit Board* (PCB) otimizados, o Autodesk Eagle oferece uma plataforma

robusta para a concepção e implementação de circuitos eletrônicos. Foi utilizado nesse projeto o Autodesk Fusion 360 (AUTODESK INC., 2017) para a modelagem 3D de componentes e estruturas físicas.

No que se refere à análise e validação de circuitos de *hardware*, o LTSPICE se destaca como uma ferramenta poderosa para simulação de circuitos. Com sua capacidade de modelar circuitos eletrônicos complexos e prever seu comportamento sob diferentes condições, o LTSPICE permite uma avaliação abrangente do desempenho e da funcionalidade dos sistemas projetados, conforme descrito em (NASCIMENTO, L; VARGAS, J. B, 2015). A combinação desses três softwares - Autodesk Eagle, Autodesk Fusion e LTSPICE - oferece uma abordagem completa e eficaz para o desenvolvimento de sistemas.

3.2 Metodologia

A metodologia de pesquisa exploratória permitirá explorar e compreender a viabilidade e a eficácia do sistema proposto, enquanto a abordagem experimental possibilitará a realização de testes e validações práticas.

3.2.1 Estudo dos recursos disponíveis do Raspberry Pi 3

- Realizar uma revisão da literatura para compreender as capacidades e funcionalidades do Raspberry Pi 3, especialmente em relação à sua compatibilidade com o sistema operacional Linux e a linguagem de programação Python.
- Explorar os recursos de GPIO (*General Purpose Input/Output*) do Raspberry Pi 3 e suas possibilidades de controle e interação com os componentes do sistema, utilizando bibliotecas e *frameworks* compatíveis com Linux/Python.

3.2.2 Modelamento do sistema de controle

- Investigar técnicas e algoritmos de controle adequados para o sistema de monitoramento.
- Desenvolver um modelo matemático do sistema de controle que relacione as variáveis de entrada (dados dos sensores) com as saídas desejadas (controle dos atuadores).
- Realizar simulações para verificar a eficácia do sistema de controle proposto.

3.2.3 Realização de testes em um protótipo de baixa potência

- Montar um protótipo do sistema de monitoramento utilizando os dispositivos Raspberry Pi 3 e o expansor de GPIO MCP23017.
- Conectar os sensores de temperatura, umidade, gás e os dispositivos de controle de acesso físico ao protótipo.
- Coletar dados e realizar testes para verificar a precisão e o desempenho do sistema de monitoramento em condições de baixa potência.

3.2.4 Desenvolvimento da interface com o usuário

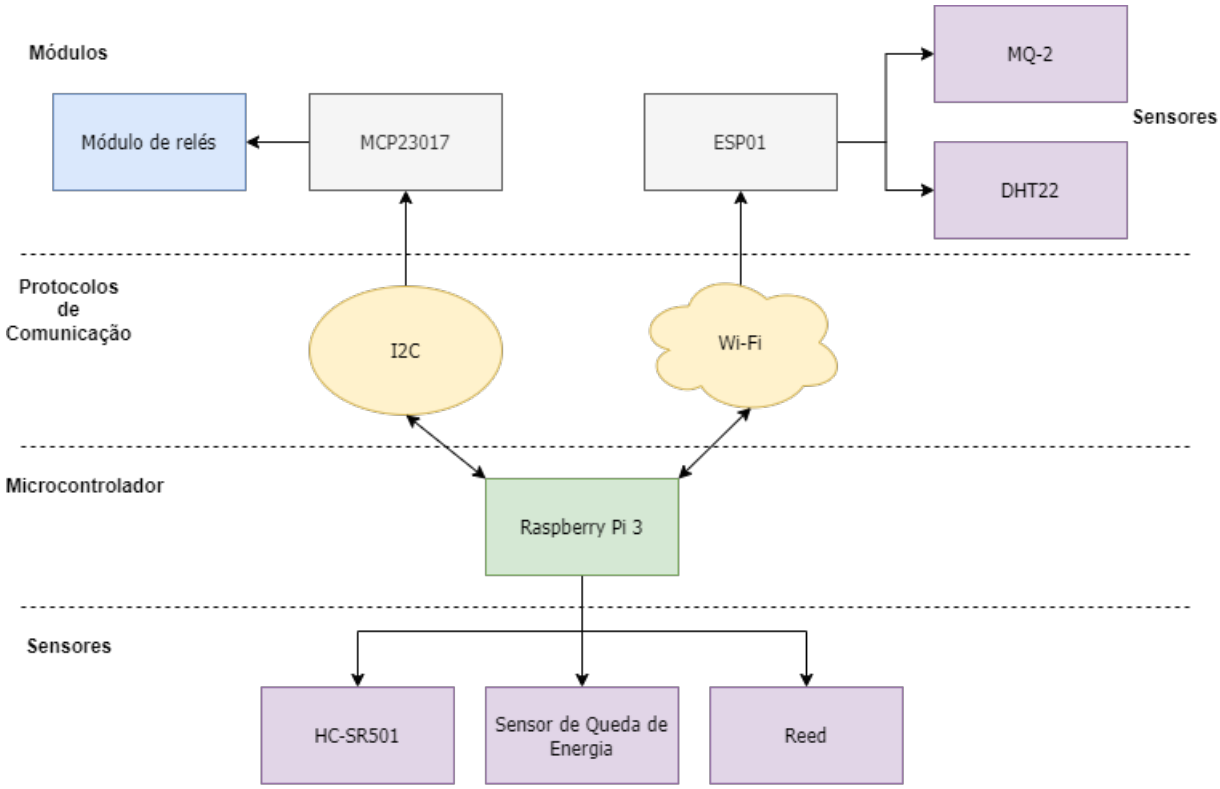
- Projetar e implementar uma interface de usuário amigável e intuitiva para visualização e gerenciamento dos dados coletados pelo sistema de monitoramento.
- Integrar a interface com o aplicativo supervisório, permitindo o acesso remoto e em tempo real aos dados e controles do sistema.

4 DESENVOLVIMENTO

4.1 Arquitetura do sistema embarcado

No diagrama de blocos apresentado na Figura 7, é ilustrada a interconexão dos principais componentes do sistema, incluindo sensores, microcontroladores, módulos de comunicação e dispositivos de controle. Por meio dessa representação visual, é possível compreender de maneira mais clara e concisa como os diferentes elementos do sistema se relacionam entre si.

Figura 7 – Diagrama de Blocos das Conexões do Sistema de Monitoramento



Fonte: Autoria própria (2024).

4.2 Descrição de Funcionamento

O sistema de monitoramento realiza suas ações com base na leitura dos sensores. Cada variável de monitoramento é lida individualmente e o microcontrolador utiliza o sistema de eventos e alarmes, além de acionar os atuadores, para garantir o funcionamento adequado do projeto. Os componentes incluídos no projeto são o sensor de umidade e temperatura, sensor de corrente, sensor de fumaça, sensor de movimento e/ou presença, e o sensor magnético de porta.

A fim de tratar esses pontos distintos, é necessário um software que processe os sinais captados e forneça as informações necessárias ao usuário, permitindo que ele esteja ciente do que está ocorrendo em seu *data center* e possa agir de forma autônoma para corrigir ou evitar comportamentos indesejados.

Os eventos dependem diretamente das leituras dos sensores. O sensor DHT22, responsável pelas medições de temperatura e umidade, possui faixas de valores específicas para um bom funcionamento. Recomenda-se uma tolerância de 10% para a temperatura e 5% para a umidade, de acordo com Caruso e Steffen (2006). Além disso, a variação da temperatura não deve exceder 1º C a cada 5 minutos, e a umidade relativa deve permanecer entre 45% e 55% ao longo de 8 horas. Se as leituras estiverem fora dessas faixas permitidas, o sistema verifica o sensor de corrente ligado junto ao ar condicionado para determinar se está em funcionamento e, em seguida, notifica o proprietário.

Quando o sensor MQ-2 detecta fumaça e/ou gás inflamável, o sistema realiza duas ações. Primeiramente, um alarme sonoro é ativado para alertar sobre a presença desses elementos perigosos e será enviado uma notificação ao usuário informando sobre a detecção. No caso de detecção de movimento pelo sensor de presença HC SR501 será verificado se o movimento ocorreu durante um horário permitido. Se sim, o sistema notifica o proprietário sobre a presença de movimento. No entanto, se o movimento for detectado em um horário não permitido, o sistema ativa o alarme sonoro para alertar sobre uma possível intrusão.

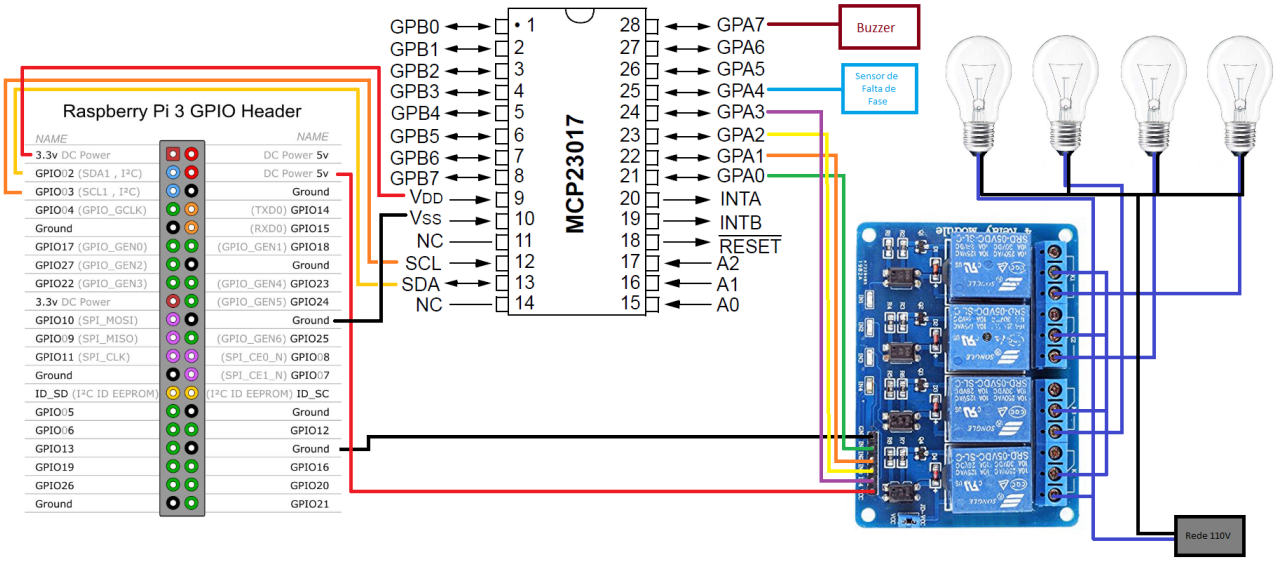
O bloco responsável por verificar a abertura da porta segue um processo semelhante, incluindo a verificação de horário. Nesse caso específico, se a abertura da porta ocorrer durante um horário permitido, duas ações adicionais são executadas. Primeiro, o sistema envia uma notificação ao usuário informando sobre a abertura da porta, em seguida, as luzes do ambiente são automaticamente ligadas, proporcionando uma iluminação adequada, sendo esta uma medida sustentável para evitar o gasto de energia.

A detecção de queda de energia é o único evento que interrompe o ciclo contínuo do processo representado no diagrama. Quando uma queda de energia é identificada, o sistema entra em ação enviando uma notificação aos usuários.

4.3 Modelagem do sistema

A Figura 8, apresenta um diagrama das conexões do MCP (Microcontrolador de Periféricos) no sistema. Este diagrama oferece uma representação visual das conexões físicas entre o MCP e os dispositivos periféricos, como sensores, atuadores e outros componentes do sistema. Essa visualização facilita a compreensão da arquitetura de *hardware* do projeto e ajuda na identificação e resolução de possíveis problemas de conexão.

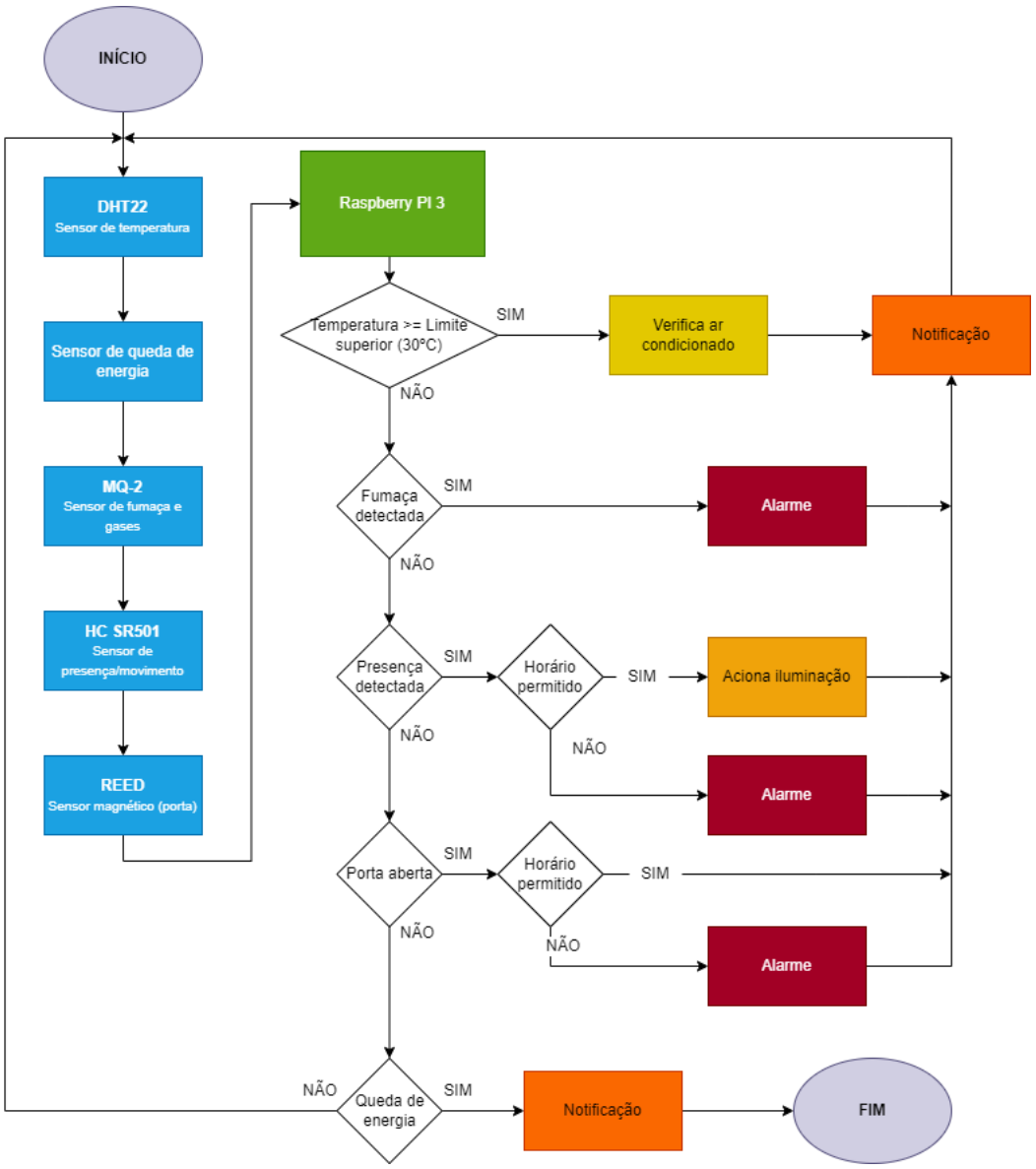
Figura 8 – Esquemático do MCP



Fonte: Autoria própria (2024).

Na Figura 9 será apresentado um fluxograma do sistema, ilustrando de forma visual o funcionamento e a interação entre os diferentes componentes e módulos do projeto. Este fluxograma servirá como um guia para compreender o fluxo de dados e as etapas executadas pelo sistema durante sua operação.

Figura 9 – Diagrama de funcionamento



Fonte: Autoria própria (2024).

4.4 Hardware

Na seção 4.4, é descrito o hardware utilizado no projeto, incluindo o sensor de falta de fase e o módulo de monitoramento ambiental. O sensor de falta de fase utiliza o componente PC817 para detectar quedas de energia, enquanto o módulo Wi-Fi com ESP-8266 integra os sensores DHT22 e MQ-2 para monitorar temperatura, umidade e concentração de gases no data center.

4.4.1 Projeto do sensor falta de fase

Propomos abaixo um circuito básico de monitoramento para quedas de energia, utilizando o componente PC817. Esse sistema fornecerá um valor de referência para a porta lógica do nosso sistema de monitoramento. O PC817 é um acoplador óptico amplamente utilizado para isolar eletricamente circuitos de controle e potência. Seu princípio de funcionamento se baseia na transferência de sinal por meio de um LED infravermelho e um fototransistor, proporcionando isolamento galvânico entre os circuitos de entrada e saída. Os conceitos básicos para a construção deste projeto são similares a um retificador monofásico de meia-onda, combinado com chaveamento lógico usando transistores.

No esquemático mostrado na Figura 10, na primeira etapa do circuito, o condutor fase da fonte V1 é conectado em série com o capacitor C1, que produz uma queda de tensão e fornece a corrente necessária para o funcionamento do LED. De acordo com o Datasheet, a tensão e corrente típicas para a polarização direta são de 1,2V e 20mA, respectivamente. Portanto, a resistência equivalente do LED é calculada como:

$$R_{\text{LED}} = \frac{V_{\text{LED}}}{I} = \frac{1{,}2\text{V}}{20\text{mA}} = 60\Omega \tag{1}$$

Ao utilizar uma tensão fase-neutro de aproximadamente 179,6Vp, com frequência de 60Hz, é possível determinar o valor ideal do capacitor a ser utilizado. Dessa forma, a impedância necessária é determinada por:

$$Z_T = \frac{V_{\text{V1}}}{I_T} = \frac{179{,}6\,\text{V}_p}{20\,\text{mA}} = 8980\,\Omega \tag{2}$$

Logo, a reatância capacitiva é calculada como:

$$X_C = \sqrt{Z^2 - R_{\text{LED}}^2} = \sqrt{8980^2 - 60^2} \approx 8979{,}79\Omega \tag{3}$$

Finalmente, o valor do capacitor é determinado por:

$$C = \frac{1}{2\pi F X_C} = \frac{1}{2\pi \times 60 \times 897{,}79} \approx 295{,}39\text{nF} \tag{4}$$

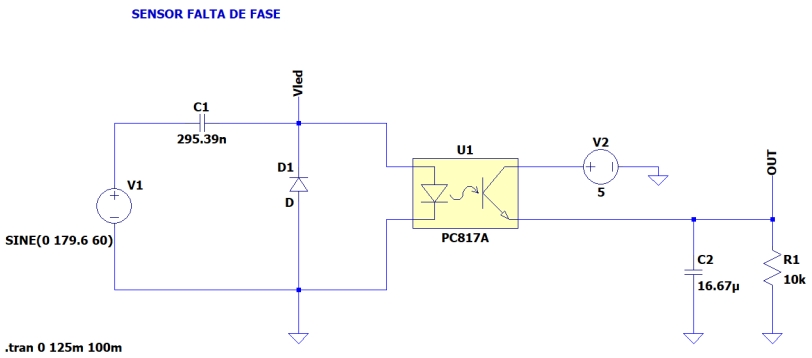
A escolha do capacitor, em contrapartida a um resistor de potência, dá-se pela dissipação da energia em reatância capacitiva ao invés de calor. O diodo D1 em antiparalelo conduz toda a tensão negativa do circuito, protegendo o LED de danos por tensões reversas. Na segunda parte, como o LED estará aceso apenas nos semiciclos positivos da fonte V1, o fototransistor interno do PC817 irá chavear a fonte V2 em uma frequência de 60Hz, que é retificada pelo capacitor C2 em conjunto com o resistor R1 para reduzir o ripple. Os cálculos são demonstrados na sequência.

Considerando um ripple de 0,5V e uma resistência de $10k\Omega$, a capacitância necessária é dada pela equação:

$$C = \frac{V_1}{V_r f R} = \frac{5V}{0{,}5 \times 60 \times 10k} \approx 16{,}67\text{uF} \tag{5}$$

Nesse caso, haverá um sinal lógico de aproximadamente 4,5V na saída do circuito.

Figura 10 – Esquemático sensor de queda de energia

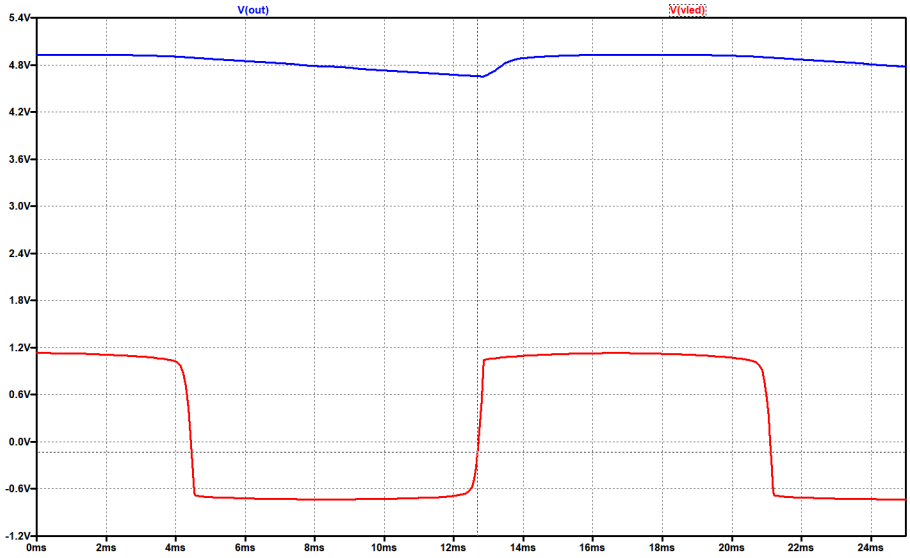


Fonte: Autoria própria (2024).

4.4.1.1 Simulação do sensor

A Figura 11 representa as formas de onda simuladas para o circuito demonstrado anteriormente na Na Figura 10, é apresentado o gráfico para a tensão em cima do LED em vermelho e o valor para a saída em azul.

Figura 11 – Simulação sensor de queda de energia



Fonte: Autoria própria (2024).

É possível observar que os valores obtidos na simulação estão próximos aos calculados para a correta operação. É importante lembrar que, este circuito foi projetado exclusivamente para operação em sistemas com alimentação monofásica.

4.4.2 Módulo Wi-Fi com ESP-8266

Neste segmento, será apresentada a implementação de um módulo de monitoramento ambiental empregando o microcontrolador ESP8266, em conjunto com os sensores DHT22 e MQ-2 que fará interface com o sistema principal na Raspberry Pi 3, este módulo pode ser aplicado em diversas partes do ambiente do *data center* para monitoramentos pontuais das váriaveis de temperatura, umidade e concentração de gases.

O módulo de monitoramento foi construído utilizando uma placa de desenvolvimento ESP-01, conectada aos sensores DHT22 e MQ-2 por meio de pinos 0 e 2, respectivamente. Também foi previsto uma circuito extra de relê que pode ser utilizado para acionamento de algum atuador, caso necessário. O módulo foi devidamente alimentado por uma fonte de 5V retificada pelo tradicional CI LM1117 de 3,3V o qual garante tensões ideais para o ESP-01 e os sensores. Se necessário, pinos de jumper trazem tensão 5V da fonte até a entrada de alimentação dos sensores.

O microcontrolador ESP8266 realiza a leitura dos sensores periodicamente, sendo esses dados processados e enviados por Wi-Fi para a interface HTML. O *firmware* que implementa a lógica citada acima foi desenvolvido com a utilização da Arduino IDE, por já possuir compatibilidade com o módulo escolhido. Para disponibilizar os dados coletados, o módulo de monitoramento utiliza a capacidade de comunicação *Wi-Fi* do ESP01 para enviar as informações para a Raspberry Pi (RPi). Lá, esses dados podem ser visualizados e analisados em tempo real por meio de uma interface web em HTML.

O esquemático completo do módulo é apresentado no Apêndice A.

4.5 Firmware

O capítulo de *firmware* deste trabalho aborda a implementação do *software* embarcado responsável pelo controle e gerenciamento do sistema de monitoramento. Nele, serão discutidas as estratégias de desenvolvimento adotadas, as linguagens e ferramentas utilizadas, além da arquitetura e organização do código-fonte. Este capítulo oferece uma visão detalhada das etapas de implementação, destacando sua relevância para o funcionamento eficiente do sistema.

4.5.1 Python e bibliotecas

Durante a implementação do código em Python para o sistema de monitoramento, foram utilizadas algumas bibliotecas de desenvolvimento. Entre elas, a SMBus (LINUX KERNEL ORGANIZATION, 2024) que desempenha um papel fundamental na comunicação I2C entre o microcontrolador e os dispositivos periférico. Essa biblioteca oferece métodos para enviar e receber dados pelo barramento I2C, facilitando a leitura e o controle desses dispositivos. Além disso, a biblioteca *requests* (REITZ, K, 2024) é essencial para realizar requisições HTTP em Python, possibilitando o envio de dados para um servidor remoto, consultas a APIs externas e até mesmo o envio de notificações por meio de serviços online. Por fim, a biblioteca AdafruitDHT (ADAFRUIT INDUSTRIES, 2024) é utilizada especificamente para a leitura de sensores de temperatura e umidade da família DHT, fornecendo métodos para obter leituras de sensores.

4.5.2 Desenvolvimento

4.5.2.1 Arquitetura do Código

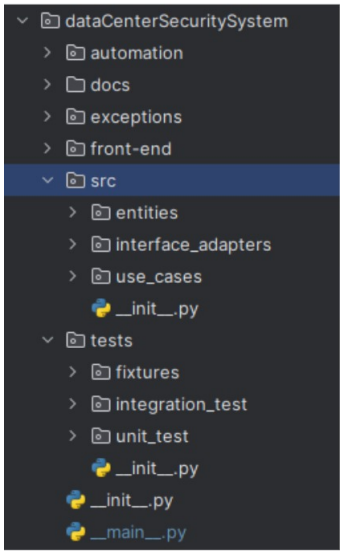
No desenvolvimento deste trabalho, a arquitetura do código foi meticulosamente planejada, seguindo as diretrizes estabelecidas no livro *Clean Architecture* de Robert C. Martin (MARTIN, 2017). Seguindo os princípios propostos pelo autor, o código fonte foi organizado de forma a promover a manutenibilidade, escalabilidade e testabilidade do sistema.

Para isso, adotou-se a divisão do código em três pastas principais: entidades, *interface adapters* e *use cases*.

As entidades representam as estruturas de dados e regras de negócio fundamentais do sistema, enquanto os adaptadores de interface são responsáveis por implementar uma adaptação dos dados entre as entidades e os casos de uso, além de integrar ambos com interfaces externas, como drivers e APIs. Por fim, os *use cases* representam os casos de uso da aplicação,

contendo a lógica de funcionamento do negócio em questão e quais serão as regras específicas da aplicação e dos servços que serão chamados em tempo de execução. Essa estruturação do código facilita a compreensão, manutenção e evolução do sistema ao longo do tempo, seguindo as melhores práticas de desenvolvimento de *software*.

Figura 12 – Arquitura do código - Pycharm

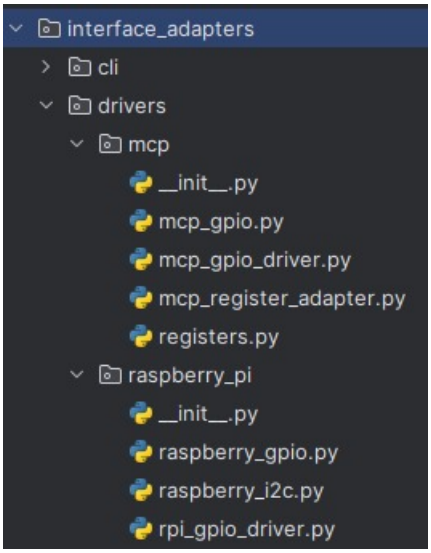


Fonte: Autoria própria (2024).

4.5.2.2 Drivers

Para fazer a comunicação entre MCP23017 e Raspberry Pi, foram desenvolvidas classes que abstraem a informação dos dispositivos e desenvolvem métodos para escrita e leitura de *bytes* de dados.

Figura 13 – Drivers - Pycharm



Fonte: Autoria própria (2024).

Sendo assim, as duas principais camadas de adaptadores de interface são as classes:

- RaspberryI2C: Classe que implementa os métodos para comunicação com o Raspberry Pi através do protocolo I2C.
- MCPGpio: Implementa um *driver* de portas programáveis de entrada e saída para se comunicar com o CI através do protocolo I2C.

Na Listagem 1 pode ser visualizado um trecho do código fonte.

Listagem 1 – Código Driver I2C

```python
from smbus2 import SMBus

from dataCenterSecuritySystem.src.interface_adapters.i2c.i2c_interface
import I2CInterface

class RaspberryI2C(I2CInterface):
    """Class that implements the methods to raspberry communicate
    by I2C protocol."""

    def __init__(self):
        """Constructor for raspberry I2C."""
        self._bus = SMBus(1)
        self._address = 0x20

    def read_byte_data(self, register: int) -> int:
        """Read data from a given register.

        Args:
            register(int): desired register for read data.

        Returns:
            int: Data read from register.
        """
        return self._bus.read_byte_data(self._address, register)

    def write_byte_data(self, register: int, value: int) -> None:
        """Write a value in given register.

        Args:
            register(int): register target that will receive the value.
            value(int): value to update data.
        """
        self._bus.write_byte_data(self._address, register, value)
```

Fonte: Autoria própria (2024).

4.5.2.3 Classe MCPGpio

A classe ‘MCPGpio‘ é responsável por fornecer métodos para configurar e controlar os pinos GPIO. No construtor da classe, o *driver* I2C é inicializado juntamente com os pinos do MCP23017 e um adaptador dos registradores do MCP. Métodos privados são utilizados para configurar os pinos e obter os estados dos registradores e a configuração correspondentes a partir de seus nomes definidos em um Enum. Já os métodos públicos permitem configurar ou ler o estado e a direção dos pinos e realizar a configuração e inicialização deles, quando o usuário julgar necessário.

4.5.2.4 *Command Line Interface*

Uma Interface de linha de comando, do inglês *Command Line Interface* (CLI) é uma forma de interação com programas ou sistemas através de texto inserido em um terminal ou prompt de comando. Os usuários digitam comandos simples, seguidos de opções e argumentos, se necessário, para executar ações específicas.

A CLI é implementada em Python utilizando o módulo *argparse* para analisar e processar os argumentos fornecidos pelo usuário. Esses argumentos incluem opções como arquivo de configuração, comando, rótulo e ação.

A estrutura do sistema permite que o usuário execute diversas operações, como configurar o estado do sistema ou executar a aplicação principal. A análise dos argumentos da linha de comando determina qual ação será tomada, simplificando assim o controle do sistema.

4.5.2.5 Adaptador de arquivos - JSON

A classe de ‘*files adapter*‘ JSON permite a leitura de um arquivo de configuração JSON e sua conversão em objetos dos tipos Python, como dicionários, inteiros, flutuantes ou listas, permitindo que os dados sejam facilmente manipulados e utilizados na aplicação. Isso simplifica a integração e o uso dos dados do arquivo JSON no desenvolvimento da aplicação e permite que o sistema seja configurado de acordo com a preferência do usuário.

4.5.2.6 Classes dos Sensores

As classes de sensores foram desenvolvidas de maneira a permitir que novos sensores sejam facilmente integrados ao sistema sem modificar o código existente, garantindo a flexibilidade e extensibilidade da solução. No construtor, é especificado o pino GPIO ao qual o sensor está conectado, permitindo a configuração da comunicação com o sensor. O método ‘read()‘ é responsável por obter os dados de detecção de movimento do sensor, configurando o pino GPIO para entrada e obtendo seu estado atual. Esse estado é então retornado como um valor

‘GPIOState‘ que pode ser do tipo alto ou baixo, indicando, por exemplo, se um movimento foi detectado ou não.

Para expandir essa classe e torná-la compatível com diferentes tipos de sensores, estratégias como a definição de uma interface genérica de sensor, o uso de herança para criar subclasses específicas para cada tipo de sensor e a injeção de dependência no construtor da classe para permitir a troca dinâmica do tipos de sensores foram utilizadas.

4.5.2.7 Serviços

As classes de serviços foram desenvolvidas como parte integrante do sistema de monitoramento, desempenhando funções específicas para garantir o funcionamento coordenado do sistema. Entre esses serviços, destaca-se o Serviço de Reset dos Pinos, responsável por restaurar os estados dos pinos GPIO para um estado inicial, assegurando que o sistema comece em uma condição conhecida e consistente. Essa medida é crucial para prevenir problemas de inicialização.

Outro serviço relevante é o Serviço de Dump dos Estados dos GPIOs, que possibilita a obtenção de um "instantâneo"dos estados atuais dos pinos GPIO, oferecendo informações sobre quais pinos estão configurados em nível alto (*HIGH*) ou baixo (*LOW*). Essa funcionalidade pode ser empregada para diagnosticar eventuais falhas de *hardware* ou monitorar o estado do sistema em tempo real, contribuindo para a manutenção proativa do sistema.

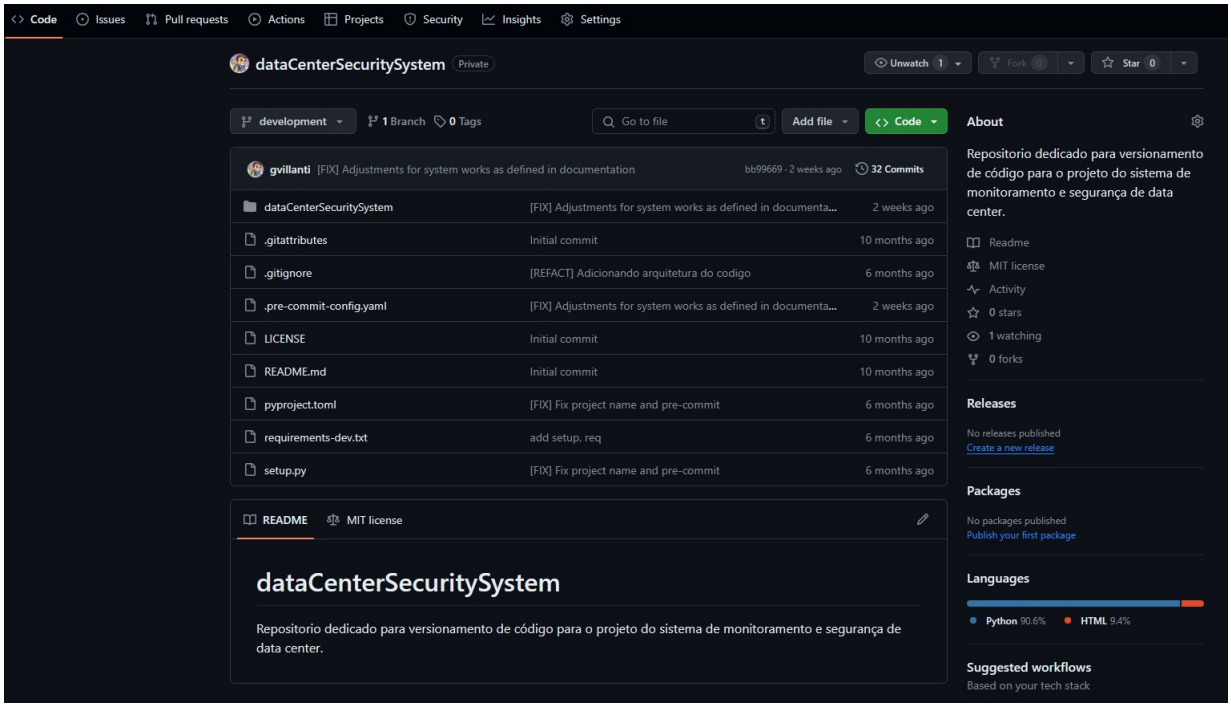
Além disso, destaca-se o Serviço Principal da Aplicação, que desempenha um papel central na coordenação das principais funcionalidades do sistema de monitoramento. Esse serviço inclui métodos dedicados a verificar a presença de movimento, monitorar a abertura de portas, detectar falhas de energia e tomar ações correspondentes com base nessas informações. Por exemplo, ele verifica de forma contínua se o sensor de movimento detecta qualquer movimento e, se houver detecção, aciona a iluminação da sala. Adicionalmente, monitora a abertura de portas e envia alertas se alguma porta for aberta fora do horário permitido. Também é responsável por detectar falhas de energia e enviar alertas em caso de interrupção no fornecimento de energia.

Esses serviços foram implementados seguindo as melhores práticas de programação e *design* de *software*, adotando padrões de codificação consistentes e documentação adequada do código para facilitar a manutenção e expansão futura do sistema. A modularidade desses serviços permite que novas funcionalidades sejam incorporadas de maneira eficiente, possibilitando a adaptação do sistema para atender a diferentes requisitos e casos de uso.

4.5.2.8 Controle de Versionamento

Para o desenvolvimento contínuo do *firmware* do projeto, foi utilizada a plataforma GitHub, criando um repositório para organização de versões. Utilizando comandos do Git para versionamento remoto do código, foi possível realizar operações padrões de controle de repositório, como *commit*, *push* e *fetch*.

Figura 14 – Repositório no GitHub



Fonte: Autoria própria (2024).

4.5.3 Testes e validação

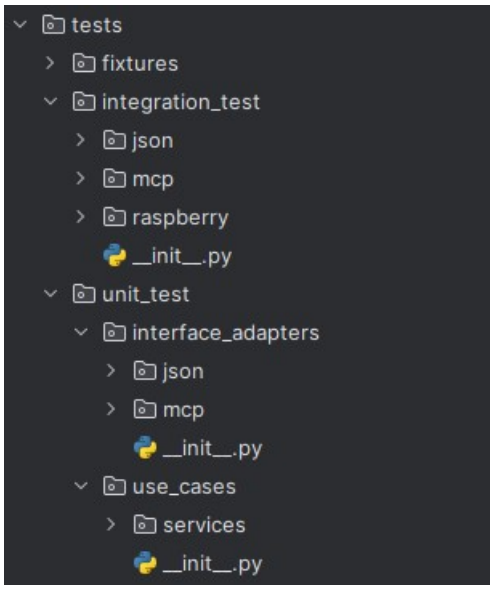
Na fase de testes e validação, foram elaborados testes unitários e testes de integração para assegurar a robustez e a eficácia do sistema desenvolvido. O plano de testes abordou todos os cenários possíveis de operação do sistema, considerando as variáveis de controle inerentes a um sistema de monitoramento. Os testes unitários avaliaram individualmente cada componente do sistema para verificar se suas funcionalidades estão alinhadas com as especificações.

Por outro lado, os testes de integração garantiram a adequada interação entre os diferentes módulos, sensores e atuadores, simulando o fluxo completo de dados e comunicação entre eles. Essa abordagem permitiu identificar e corrigir possíveis falhas, garantindo que o sistema atenda aos requisitos estabelecidos e proporcione um ambiente de monitoramento confiável e eficiente.

Para simular variações de temperatura, o sensor foi estrategicamente posicionado para realizar leituras em condições de exposição direta ao sol e ao ar condicionado. No que diz

respeito à detecção de gases e líquidos inflamáveis, um frasco contendo álcool foi disposto nas proximidades do equipamento, visando avaliar a sensibilidade e precisão de detecção. Além disso, testes adicionais foram conduzidos, incluindo a utilização de sensores de presença para identificar movimentos nas imediações do sistema e a simulação do movimento de abertura de portas mediante a aproximação de um ímã. Esses procedimentos possibilitaram a simulação efetiva dos eventos que serão controlados pelo sistema durante o seu funcionamento em regime permanente, contribuindo para a validação e aperfeiçoamento do projeto.

Figura 15 – Repositório de Testes



Fonte: Autoria própria (2024).

Na Listagem 2 é apresentado um exemplo de teste de integração que valida a classe MCPGpio.

Listagem 2 – Exemplo de código de teste de integração

```
try:
    from src.interface_adapters.drivers.raspberry_pi.raspberry_i2c
    import RaspberryI2C
except ModuleNotFoundError:
    raise pytest.importorskip(
    "SMBus", reason="Skipped due to not supported on Windows OS."
    )


def test_set_all_output(
raspberry_i2c: RaspberryI2C,
gpio_integration_test: MCPGpio
):
    """Tests the MCP set_all_output method"""
    # Arrange
    expected_result = 0

    # Act
    gpio_integration_test.set_all_output()
    result = raspberry_i2c.read_byte_data(MCPRegistersAddresses.IODIRB)

    # Assert
    assert expected_result == result


def test_set_pin_state_a(
raspberry_i2c: RaspberryI2C,
gpio_integration_test: MCPGpio
):
    """Test the MCP set_pin_state method"""
    # Arrange
    expected_result = 7

    # Act
    gpio_integration_test.pin_cleanup()
    gpio_integration_test.set_all_output()
    gpio_integration_test.set_pin_state("MCP_GPIO_A0", GPIOState.HIGH)
    gpio_integration_test.set_pin_state("MCP_GPIO_A1", GPIOState.HIGH)
    gpio_integration_test.set_pin_state("MCP_GPIO_A2", GPIOState.HIGH)
    gpio_integration_test.set_pin_state("MCP_GPIO_A3", GPIOState.LOW)
    gpio_integration_test.set_pin_state("MCP_GPIO_A4", GPIOState.LOW)

    i2c_result = raspberry_i2c.read_byte_data(MCPRegistersAddresses.GPIOA)

    # Assert
    assert i2c_result == expected_result
```

Fonte: Autoria própria (2024).

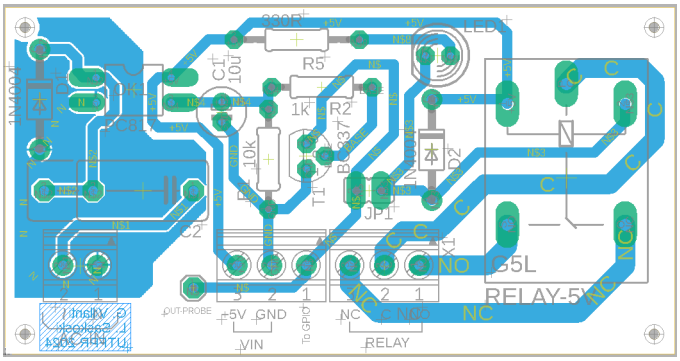
5 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os resultados obtidos com o desenvolvimento do trabalho, divido em seções de hardware e interface com o usuário e terminando com a seção da protóripo da protótipo que faz a integração entre as duas anteriores.

5.1 Protótipo de Hardware

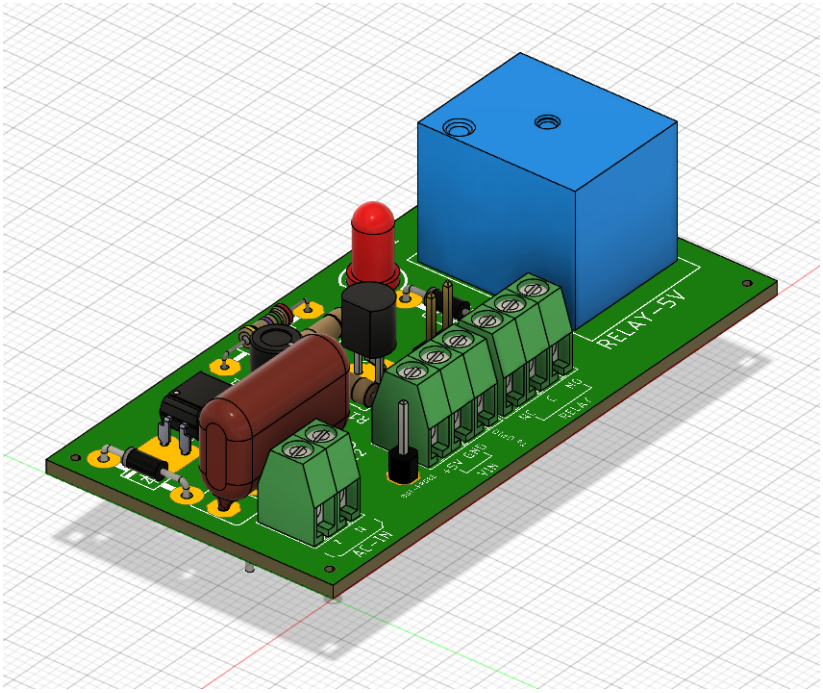
As placas de circuito impresso para os dispositivos definidos na seção 6.2 foram projetadas no software Autodesk Eagle, e reproduzidos na prática em fenolite através do processo de transferência térmica e corrosão com Percloreto de Ferro (FeCl3). Para melhorar a percepição e garantir o melhor posicionamento de todos os componentes envolvidos nos projetos, também foram desenvolvidos modelos 3D para estes dispositovos na plataforma Autodesk Fusion 360, Abaixo apresentam-se os resultados obtidos.

Figura 16 – Desenho da PCI para o sensor de queda de energia



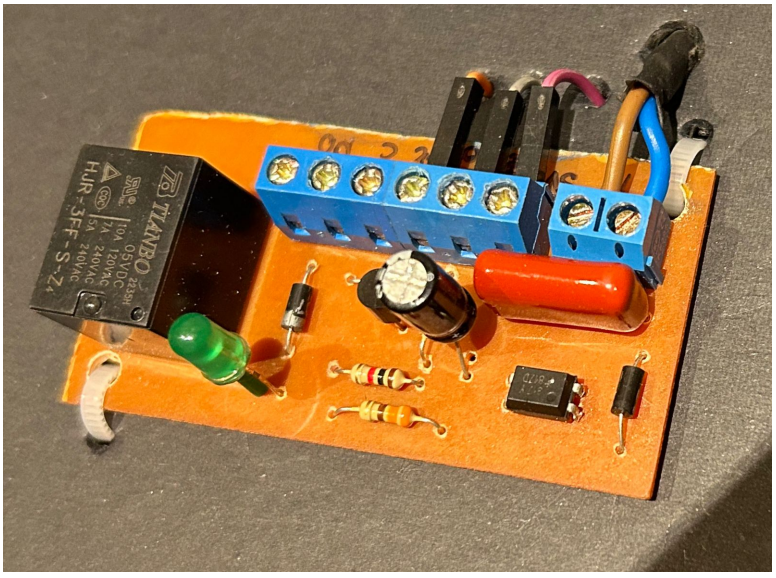
Fonte: Autoria própria (2024).

Figura 17 – Modelo 3D para o sensor falta de fase



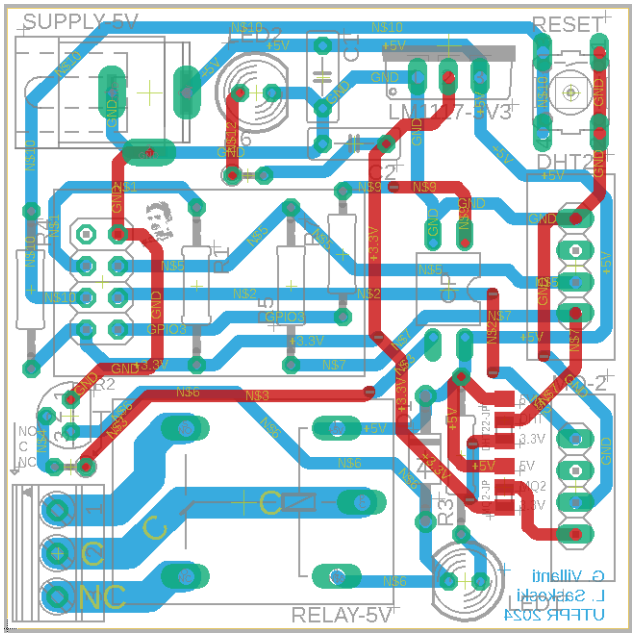
Fonte: Autoria própria (2024).

Fotografia 1 – Sensor de queda de energia montado



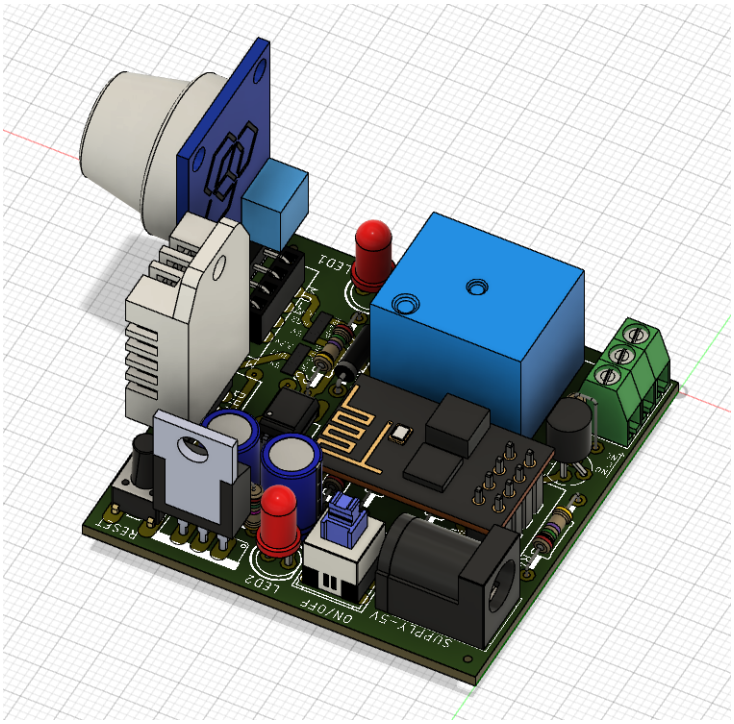
Fonte: Autoria própria (2024).

Figura 18 – Desenho da PCI para o módulo Wi-Fi



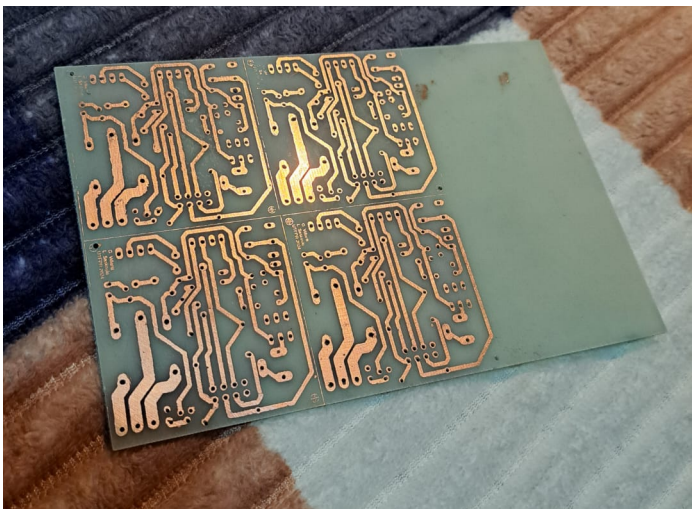
Fonte: Autoria própria (2024).

Figura 19 – Modelo 3D para o módulo Wi-Fi



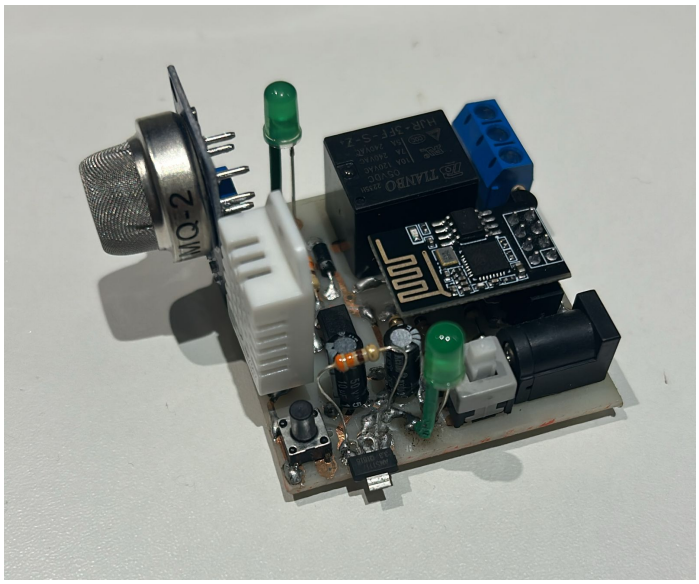
Fonte: Autoria própria (2024).

Fotografia 2 – PCI corroida módulo Wi-Fi



Fonte: Autoria própria (2024).

Fotografia 3 – Sensor módulo Wi-Fi



Fonte: Autoria própria (2024).

5.2 Interface com usuário

5.2.1 Página HTML

Inicialmente, são incluídas as bibliotecas necessárias para a comunicação *WiFi* e interação com os sensores DHT22 (para temperatura e umidade) e MQ2 (para gás). Em seguida, são definidos os pinos GPIO aos quais os sensores estão conectados e são instanciados objetos para interagir com esses sensores.

No método *setup()*, o código realiza a inicialização básica, como a configuração da comunicação serial, inicialização do sensor DHT22 e configuração do pino do relé como saída.

Além disso, é estabelecida a conexão do ESP8266 à rede *WiFi*. No loop principal (*loop()*), o código verifica se há clientes conectados ao servidor *web*. Se um cliente estiver conectado, o código analisa a requisição HTTP e executa a função correspondente com base no caminho da requisição.

As funções de manipulação de requisições incluem a resposta com os dados dos sensores em formato HTML ou JSON, dependendo do tipo de requisição. Por exemplo, a função *handleData()* retorna uma página HTML com as leituras dos sensores de temperatura e gás, enquanto as funções *handleTemperature()* e *handleGas()* respondem com objetos JSON contendo as leituras individuais de temperatura e a presença de gás inflamável, respectivamente. Adicionalmente, o código atualiza o estado do relé com base nas leituras do sensor de gás. Se gás inflamável for detectado, o relé é acionado para ativar um dispositivo de segurança

5.2.2 Robô mensageiro: BOT Telegram

Com base na implementação do projeto, foi desenvolvido um robô mensageiro utilizando a linguagem de programação Python. Esse robô mensageiro tem como função principal enviar alertas instantâneos para os interessados sempre que ocorrerem eventos detectados pelo sistema. Esses eventos incluem, entre outros, movimentos fora do horário permitido, intrusões não autorizadas ou detecções de leituras críticas dos sensores.

Por meio de uma integração eficiente com o aplicativo de mensagens Telegram, os usuários são notificados em tempo real sobre essas ocorrências, permitindo uma resposta rápida e eficaz diante de situações potencialmente problemáticas. Essa abordagem proporciona um mecanismo de alerta proativo e automatizado, que é fundamental para garantir a segurança e a integridade do ambiente monitorado pelo projeto.

A implementação da classe pode ser observado na Listagem 3, que representa a classe do BOT de Telegram.

Listagem 3 – Classe TelegramBOT

```python
from datetime import datetime

import requests


class TelegramBot:
    """Class for interacting with the Telegram Bot API and
    sending messages to a specific group or user."""

    def __init__(self):
        self.chat_id = "personal_chat_id"
        self.message = ""
        self.time_now = datetime.now().strftime("%d/%m/%Y - %H:%M:%S: ")
        self.token = "personal_token_hash"

    def last_chat_id(self, token: str) -> str:
        try:
            url = "https://api.telegram.org/bot{}/getUpdates".format(token)
            response = requests.get(url)
            if response.status_code == 200:
                json_msg = response.json()
                for json_result in reversed(json_msg["result"]):
                    message_keys = json_result["message"].keys()
                    if ("new_chat_member" in message_keys)

                    or

                    ("group_chat_created" in message_keys):
                        return json_result["message"]["chat"]["id"]
                print("No group found")
            else:
                print("Response failed, status code: {}".format(
                response.status_code
                    )
                )
        except Exception as e:
            print("Error in getUpdates:", e)

    def send_message(self, message: str) -> None:
        try:
            data = {"chat_id": self.chat_id, "text": self.time_now + message}
            url = "https://api.telegram.org/bot{}/sendMessage".format(
            self.token
            )
            requests.post(url, data)
        except Exception as e:
            print("Error in sendMessage:", e)
```

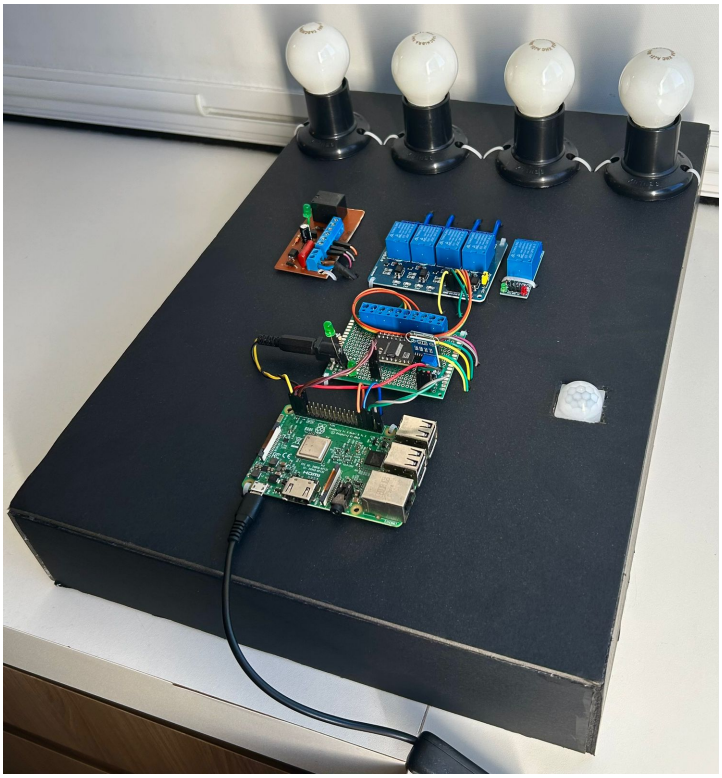
Fonte: Autoria própria (2024).

5.3 Montagem do protótipo

Durante o desenvolvimento do projeto, foi elaborado um protótipo (conforme mostrado na Foto 4), o qual distribui fisicamente todos os dispositivos utilizados. Esses dispositivos foram instalados conforme especificado no diagrama de funcionamento.

O protótipo possui um formato de caixa, o que proporciona um visual mais organizado. Além disso, esse formato permite uma melhor disposição dos cabos elétricos, que fazem a interconexão entre os dispositivos, mantendo-os escondidos por trás da estrutura.

Fotografia 4 – Protótipo montado



Fonte: Autoria própria (2024).

Em conformidade com a sequência apresentada na Fotografia 5, observa-se um conjunto de quatro lâmpadas que simulam a iluminação do ambiente do data center. Logo abaixo, à esquerda, encontra-se o sensor de queda de energia, conforme ilustrado na Fotografia 1. À direita do sensor, está posicionado o módulo de relés de 4 canais. Adicionalmente, encontra-se outro módulo de um canal, encarregado da ativação do buzzer, que por sua vez simula a sirene de alarme do ambiente. Na posição central, encontra-se o expansor de GPIO, enquanto à sua direita, está localizado o sensor de presença. Progredindo, são identificados a Raspberry Pi e o módulo Wi-Fi.

No decorrer dos testes realizados na protótipo, todos os componentes funcionaram de acordo com o esperado, validando a eficácia do sistema de monitoramento proposto. Esse resultado demonstra a robustez da solução desenvolvida, pois todos os eventos previstos foram detectados e os alarmes acionados corretamente, garantindo a segurança e a integridade do ambiente simulado do *data center*.

Fotografia 5 – Vista frontal do protótipo

Fonte: Autoria própria (2024).

A Tabela 1 mostra a precificação dos componentes utilizado na montagem de todos as partes da protótipo.

Tabela 1 – Custos do protótipo

Material	Qtd.	R$ und.	Totais
Resistor 220R	1	R$ 0,12	R$ 0,12
Resistor 100R	2	R$ 0,12	R$ 0,24
Resistor 10k	2	R$ 0,12	R$ 0,24
Resistor 1k	2	R$ 0,12	R$ 0,24
Resistor 300R	3	R$ 0,12	R$ 0,36
Capacitor de Poliéster 300nF	1	R$ 0,50	R$ 0,50
Diodo 1N7007	3	R$ 0,21	R$ 0,63
LED	3	R$ 0,24	R$ 0,72
Transistor BC337	2	R$ 0,48	R$ 0,96
Capacitor Eletrolítico 20uF	1	R$ 1,16	R$ 1,16
Borne KRE 2T	1	R$ 1,33	R$ 1,33
Conector Jack P4 Fêmea	2	R$ 0,86	R$ 1,72
Push Button 6 Pinos com Retenção	1	R$ 1,77	R$ 1,77
Capacitor Eletrolítico 10uF	2	R$ 0,99	R$ 1,98
Conector Header 2x5	1	R$ 2,31	R$ 2,31
PC817	2	R$ 1,33	R$ 2,66
Placa de Fenolíte	1	R$ 2,90	R$ 2,90
Barra de Pinos Fêmea	2	R$ 1,50	R$ 3,00
Módulo Buzzer	1	R$ 4,46	R$ 4,46
LM1117 3.3V	1	R$ 4,46	R$ 4,46
Conector Jack P4 Macho	2	R$ 2,38	R$ 4,76
Placa Perfurada	1	R$ 5,90	R$ 5,90
Módulo Relê 1 Canal	1	R$ 8,46	R$ 8,46
Relê 5V	2	R$ 4,28	R$ 8,56
Borne KRE 3T	5	R$ 1,85	R$ 9,25
Placa de Fibra Dupla Face	1	R$ 10,85	R$ 10,85
Refil de Cola Quente	1	R$ 11,40	R$ 11,40
Placa Foam	1	R$ 17,90	R$ 17,90
Cinta Plástica	1	R$ 17,90	R$ 17,90
Sensor MQ-2	1	R$ 18,03	R$ 18,03
Bocais para Lâmpada	4	R$ 5,50	R$ 22,00
Jumpers Diversos	1	R$ 22,72	R$ 22,72
Lâmpada	4	R$ 5,69	R$ 22,76
Sensor DHT22	1	R$ 22,90	R$ 22,90
Módulo Relê 4 Canais	1	R$ 23,66	R$ 23,66
Módulo MCP23017	1	R$ 49,50	R$ 49,50
Raspberry Pi 3 Model B+	1	R$ 379,00	R$ 379,00
		Total (R$)	687,31

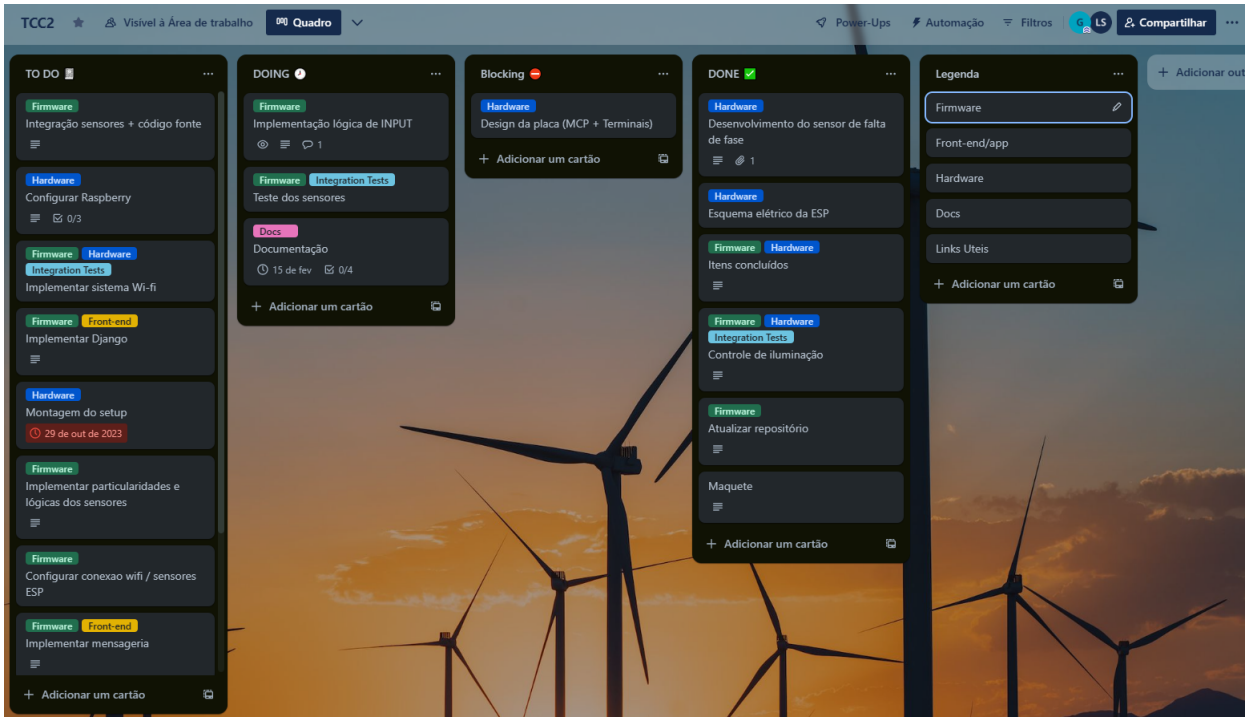
Fonte: Autoria própria (2024).

6 GESTÃO DE PROJETO

6.1 Gerenciamento de tarefas

No decorrer deste projeto, optou-se pela utilização do aplicativo **Trello** como uma ferramenta central para a organização do *backlog* de atividades e o acompanhamento do progresso do projeto. Através dessa plataforma, foi possível dividir as tarefas em etapas distintas e atribuir responsabilidades de forma clara entre os membros da equipe. Além disso, o Trello facilitou a definição de prazos para cada atividade, garantindo que os requisitos do projeto fossem atendidos dentro do cronograma estabelecido. Com uma interface intuitiva e recursos de colaboração em tempo real, o Trello se mostrou uma escolha eficaz para a gestão ágil do projeto, promovendo a transparência e a eficiência no processo de desenvolvimento.

Figura 20 – Print da tela do aplicativo Trello



Fonte: Autoria própria (2024).

6.2 Melhorias do Projeto

Após uma análise geral dos resultados, foram identificadas algumas áreas que podem ser aprimoradas para otimizar o desempenho e funcionalidades adicionais que podem ser implementadas para ampliar a eficácia do protótipo, tornando-o um produto competitivo no mercado. Algumas dessas melhorias incluem:

- Desenvolvimento de uma interface de usuário mais intuitiva e personalizável: uma melhoria significativa seria a implementação de um site com *login*, onde os usuários autorizados teriam acesso às leituras dos sensores em tempo real. Isso permitiria uma visualização mais conveniente e acessível das informações coletadas, possibilitando uma análise mais detalhada do ambiente do *data center*.

- Implementação do Banco de Capacitores para Controlar Situações de Queda de Energia: para lidar com situações de queda de energia, uma melhoria essencial seria a implementação de um banco de capacitores. Esse componente permitiria o armazenamento de energia para uso emergencial, garantindo a continuidade das operações críticas do *data center* durante falhas de energia transitórias.

- Integração com sistemas de monitoramento de segurança adicionais, como câmeras de vigilância e sistemas de detecção de incêndio, para fornecer uma visão mais abrangente da segurança do *data center*.

- *Machine learning*: Implementação de algoritmos avançados de análise de dados para identificar padrões e anomalias nas leituras dos sensores, permitindo uma detecção mais precisa de eventos e uma resposta mais rápida a possíveis ameaças.

- Implementação de um sistema de HVAC (*heating, ventilation, and air conditioning*)

7 CONCLUSÃO

Este estudo abordou de maneira abrangente a concepção, desenvolvimento e implementação de um sistema de monitoramento para data centers, visando atender às necessidades de operação, segurança e eficiência energética. Utilizando sensores de temperatura, umidade, gás e controle de acesso físico, integrados à Raspberry Pi 3 e ao expansor de GPIO MCP23017, o sistema coleta dados relevantes em tempo real e automatiza decisões para garantir o funcionamento adequado do ambiente monitorado.

A metodologia empregada combinou pesquisa exploratória e experimental, permitindo a compreensão das capacidades dos dispositivos utilizados e a validação prática do sistema desenvolvido. Interfaces de usuário intuitivas foram desenvolvidas, tanto por meio de páginas HTML quanto por um robô mensageiro via Telegram, facilitando o acesso e a interação com os dados coletados em tempo real.

Os resultados experimentais confirmaram a eficácia do sistema, com a montagem bem-sucedida do protótipo e testes que comprovaram o funcionamento adequado de todos os componentes. A integração entre hardware e software foi essencial para o desempenho satisfatório, destacando-se a robustez e a modularidade proporcionadas pela escolha dos dispositivos e plataformas de desenvolvimento.

Em síntese, não apenas foram alcançados os objetivos de desenvolver um sistema abrangente de monitoramento, mas também uma solução sólida e escalável, adaptável para diversas aplicações de automação e monitoramento em ambientes industriais e comerciais. O sistema desenvolvido representa uma ferramenta importante para profissionais da área, contribuindo significativamente para a segurança, eficiência e disponibilidade dos recursos físicos em data centers.

REFERÊNCIAS

ADAFRUIT INDUSTRIES. **Adafruit Python DHT Sensor Library**. 2024. https://github.com/adafruit/Adafruit_Python_DHT.

AOSONG. **Digital-output relative humidity temperature sensor/module DHT22 (DHT22 also named as AM2302)**. Guangzhou, China, s.d. Disponível em: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. Acesso em: 13 mar. 2024.

AUTODESK INC. **Basic API Concepts**. 2017. Disponível em: https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-D93DF10F-4209-4073-A2A0-4FA8788C8709.

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

FILHO, M. F. **Gestão da Infraestrutura do Datacenter: livro digital**. Palhoça: Unisul Virtual, 2016.

FRANCHI, C. M. **Instrumentação de Processos Industriais - Princípios e Aplicações**. E-book. [Digite o Local da Editora]: Editora Saraiva, 2015. Acesso em 25 de março de 2024. ISBN 9788536519753. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536519753.

FUTURE PUBLISHING LTD. **Raspberry Pi The Complete Manual**. Richmond House, 33 Richmond Hill, Bournemouth, Dorset BH2 6EZ: FUTURE PUBLISHING LTD, 2016. Eighth Edition.

HANWEI. **TECHNICAL DATA MQ-2 GAS SENSOR**. Zhengzhou, China, s.d. Disponível em: https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf. Acesso em: 13 mar. 2024.

JETBRAINS. **PyCharm Quick Start Guide**. 2019. Disponível em: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.

LAMBERT, K. A. **Fundamentos de Python: primeiros programas**. [*S.l.*]: Cengage Learning Brasil, 2022. Ebook. ISBN 9786555584301.

LINUX KERNEL ORGANIZATION. **Python SMBus Library**. 2024. https://www.kernel.org/doc/html/latest/i2c/smbus-protocol.html.

MARTIN, R. C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Upper Saddle River, NJ: Prentice Hall, 2017. ISBN 978-0-13-449416-6.

MASCHIETTO, L. G. *et al.* **Arquitetura e Infraestrutura de IoT**. [Digite o Local da Editora]: Grupo A, 2021. Ebook. ISBN 9786556901947.

MATTEDE, H. **Relé falta de fase – O que é e como funciona!** s.d. Disponível em: https://www.mundodaeletrica.com.br/rele-falta-de-fase-o-que-e-como-funciona/. Acesso em: 13 mar. 2024.

MICROCHIP. **MCP23017/MCP23S17: 16-Bit I/O Expander with Serial Interface**. Arizona, Estados Unidos da América, 2005.

NASCIMENTO, L; VARGAS, J. B. **Introdução à Simulação de Circuitos Eletrônicos Analógicos com Software LTspice**. [*S.l.*: *s.n.*], 2015. Novembro de 2015.

REITZ, K. **Requests: HTTP for Humans**. 2024. https://requests.readthedocs.io/en/latest/.

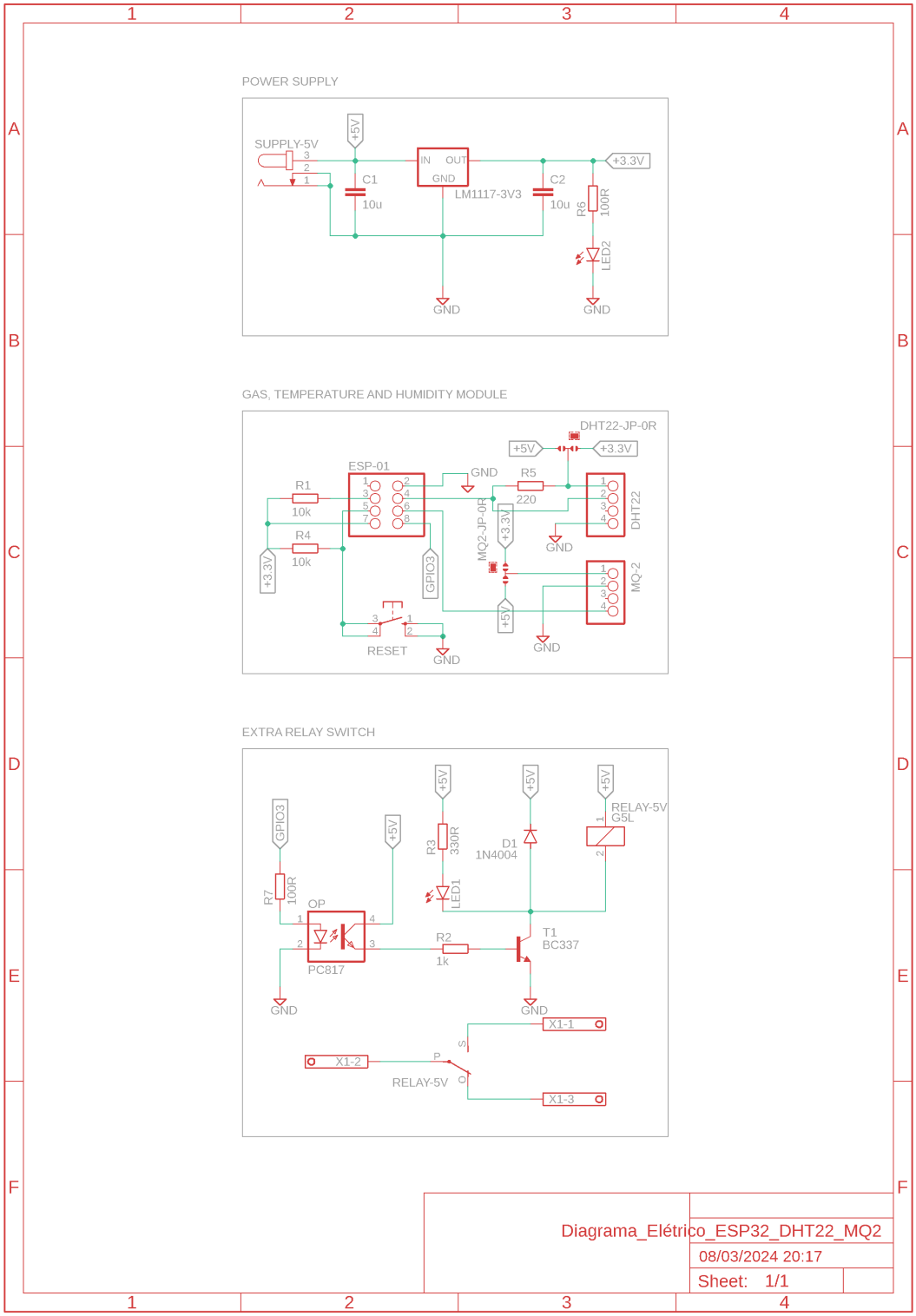
SÊMOLA, M. **Gestão de Segurança da Informação**. Rio de Janeiro: Campus, 2003.

STALLINGS, W. **Wireless Communications & Networks**. [*S.l.*]: Pearson; 2nd edition (October 3, 2013), 2013. Livro-texto.

VALDEZ, J.; BECKER, J. **Understanding the I2C Bus**. Dallas, Texas, 2015.

APÊNDICE A – Esquemático do Módulo Wi-Fi

Figura 21 – Esquemático do módulo Wi-Fi



Fonte: Os Autores.

APÊNDICE B – Arquivo de configuração JSON

Listagem 4 – Arquivo de configuração: pinos MCP23017

```
[
    {
        "label":"Lampada_4",
        "pin":"MCP_GPIO_A4",
        "pin_mode":"0"
    },
    {
        "label":"Lampada_3",
        "pin":"MCP_GPIO_A3",
        "pin_mode":"0"
    },
    {
        "label":"Lampada_2",
        "pin":"MCP_GPIO_A2",
        "pin_mode":"0"
    },
    {
        "label":"Lampada_1",
        "pin":"MCP_GPIO_A1",
        "pin_mode":"0"
    },
    {
        "label":"Buzzer",
        "pin":"MCP_GPIO_A0",
        "pin_mode":"0"
    }
]
```

Fonte: Autoria própria (2024).