FIAP - Faculdade de Informática e Administração Paulista

Global Solution - GreenBytes

👨‍🎓 Integrantes:

- Bruno Conterato
- Willian Pinheiro Marques
- Roberto Besser
- Ludimila Vitorino

👩‍🏫 Professores:

Tutor(a)

- Lucas Gomes Moreira

Coordenador(a)

- André Godoi

📜 Descrição

O GreenBytes desenvolveu um sistema inteligente para otimizar o consumo de energia residencial, integrando Inteligência Artificial (IA), Internet das Coisas (IoT), e análise de dados com Python e R, além de um banco de dados relacional (PostgreSQL). Nosso projeto abrange múltiplos aspectos da eficiência energética, desde a coleta de dados em tempo real até a previsão de consumo e a geração de relatórios personalizados.

AIC (Artificial Intelligence Challenges): Utilizamos o artigo fornecido como base para projetar um sistema de previsão de consumo energético residencial. Este sistema, ainda a ser implementado plenamente, utilizará dados históricos e em tempo real para prever o consumo futuro e otimizar o uso de eletrodomésticos. A previsão de consumo permitirá ações proativas para reduzir custos e promover a sustentabilidade. Estimamos uma economia de pelo menos 15% no consumo total, com impacto positivo no conforto e na vida útil dos equipamentos.

AICSS (Artificial Intelligence with Computer Systems and Sensors): Desenvolvemos um sistema de iluminação inteligente com um ESP32, simulando um sistema completo de monitoramento da iluminação interna e externa. Utilizando sensores de luminosidade (LDR) e de presença (PIR) simulados no Wokwi, o sistema controla o brilho dos LEDs, garantindo iluminação mínima à noite (para segurança) e regulando o brilho baseado na luminosidade e na presença detectada. O código está disponível no repositório. Um vídeo demonstrativo (link no README principal) mostra o funcionamento da solução no Wokwi. Os impactos positivos incluem a economia de energia e o aumento do conforto, enquanto os negativos são a dependência de sensores e a necessidade de calibração.

SCR (Statistical Computing with R): Analisamos o banco de dados de projetos de eficiência energética da ANEEL para gerar insights sobre estratégias eficazes e identificar lacunas no programa. Utilizamos R para análise exploratória, incluindo estatísticas descritivas, gráficos e identificação de outliers. Identificamos tendências, como a alta eficácia de projetos no setor público e oportunidades de expansão para outros setores. Nossos resultados estão disponíveis no relatório `src/SCR/eda.md`.

CDS (Cognitive Data Science): Criamos um banco de dados relacional no PostgreSQL para armazenar dados de consumo energético. O schema dimensional facilita a análise de tendências e a criação de pipelines de dados, permitindo a filtragem e análise de dados de demanda de energia elétrica e consumo per capita. A estrutura do banco de dados e os scripts SQL são descritos detalhadamente em `src/CDS/README.md`.

CTWP (Computational Thinking with Python): Desenvolvemos uma aplicação Python com interface gráfica (Tkinter) para o monitoramento em tempo real do consumo de energia. A interface simula o consumo de diversos dispositivos, exibindo dados em gráficos interativos e calculando o custo estimado. A aplicação também grava os dados simulados em nosso banco de dados PostgreSQL. O código está disponível em `src/CTWP/main.py`.

Ir Além: Integramos as soluções AICSS, CDS e SCR com a aplicação Python, consolidando um sistema completo. Um vídeo (link no README principal) demonstra a integração e a funcionalidade completa do sistema, incluindo a leitura de dados simulados do AICSS, a interação com o banco de dados CDS e a análise com R.

ℹ️ Mais informações

- Link youtube AICSS - Sensores IOT: https://youtu.be/yVlZ5D9Tw1k
- Link youtube Integração: https://youtu.be/meYFQyx6YCE
- Link Github Repositório (Códigos + Documentação): https://github.com/brunoconterato/fiap_global_solutions_2024

📁 Estrutura de pastas

```
.
├── assets      # Pasta com imagens do projeto.
│   ├── logo-fiap.png  # Logotipo da FIAP.
│   └── logo.jpeg      # Logotipo do grupo GreenBytes.
├── checklist.md # Checklist de atividades da Global Solutions.
├── README.md    # Arquivo principal com descrição geral do projeto.
├── requirements.txt # Lista de dependências Python (não descrito na atividade).
└── src          # Pasta com código-fonte e documentação por módulo.
    ├── AIC      # Módulo relacionado ao desafio de IA.
    │   └── README.md # Descrição do módulo AIC, contextualização, objetivos, desafios e resultados esperados.
    ├── AICSS     # Módulo referente ao sistema de iluminação com ESP32.
    │   ├── assets # Imagens do AICSS.
    │   │   ├── equatorial_cost_19_11_2024.png  # Print do custo da energia utilizado nos cálculos.
    │   │   └── sensors_diagram.png           # Diagrama dos sensores.
    │   ├── diagram.json  # Diagrama do circuito do Wokwi em JSON.
    │   ├── docs      # Documentação do módulo AICSS.
    │   │   ├── other  # Outros documentos do AICSS.
    │   │   │   ├── cost_estimation_external_illunination_system.md # Estimativa de custos do sistema de
iluminação externo.
    │   │   │   ├── cost_estmation_internal_illumination_system.md # Estimativa de custos do sistema de
```

```
iluminação interno.
    │   │   │   └── sensor_diagram.md  # Diagrama de componentes e descrição do sistema de sensores.
    │   │   ├── report_external_illumination_system.md # Relatório detalhado do sistema de iluminação externo.
    │   │   └── report_internal_illumination_system.md # Relatório detalhado do sistema de iluminação interno.
    │   ├── platformio.ini  # Arquivo de configuração do PlatformIO.
    │   ├── src          # Código-fonte do ESP32.
    │   │   └── sketch.ino # Código do ESP32 para controle de iluminação.
    │   ├── wokwi-project.txt # Link do projeto Wokwi.
    │   └── wokwi.toml    # Configurações do Wokwi.
    ├── CDS        # Módulo referente ao banco de dados.
    │   ├── assets     # Imagens do CDS.
    │   │   └── DER.png  # (Não descrito na atividade).
    │   ├── data       # Dados brutos utilizados para o CDS.
    │   │   ├── Anuario-Dicionario-de-Dados.pdf # Dicionário de dados do anuário estatístico.
    │   │   ├── CONSUMO MENSAL DE ENERGIA ELÉTRICA POR CLASSE.xls # Dados de consumo.
    │   │   └── Dados brutos.xlsx # Dados brutos.
    │   ├── README.md # Descrição do módulo CDS, objetivos, dados disponíveis, tecnologias e entregáveis
esperados.
    │   └── src          # Scripts SQL para criação e configuração do banco de dados.
    │       ├── create_pipeline_views.sql  # Script SQL para criação das views.
    │       └── initialize_database.sql # Script SQL para criação das tabelas.
    ├── CTWP       # Módulo referente à aplicação em Python.
    │   ├── assets     # Imagens do CTWP.
    │   │   └── interface.png  # Imagem da interface gráfica.
    │   ├── README.md # Descrição do módulo CTWP, objetivos, funcionalidades, tecnologias e como usar.
    │   ├── script     # Scripts do CTWP.
    │   │   └── initialize_db.sql # Script SQL para inicializar banco de dados (usuário, database e tabela)
    │   └── src          # Código-fonte da aplicação em Python.
    │       └── main.py  # Aplicação principal em Python com interface gráfica.
    ├── IrAlem     # Módulo referente à integração completa do sistema.
    │   ├── iot_data  # Dados simulados do sistema IoT.
    │   │   └── data.csv # Dados simulados para o sistema AICSS.
    │   ├── README.md  # Descrição da integração completa e o fluxo de dados entre os módulos.
    │   └── src          # Código para a integração dos módulos.
    │       ├── eda_files # Arquivos de saída do EDA (R).
    │       │   ├── eda_13_0.png  # Gráfico de barras de tempo de uso por dispositivo.
    │       │   ├── eda_15_0.png  # Histograma do consumo de energia.
    │       │   ├── eda_17_0.png  # Gráfico de barras do consumo de energia por dispositivo.
    │       │   ├── eda_19_0.png  # Distribuição dos dispositivos.
    │       │   └── eda_21_0.png  # Análise da relação entre consumo e tempo.
    │       ├── eda.ipynb    # Notebook Jupyter para EDA (R).
    │       ├── eda.md       # Relatório da análise exploratória de dados.
    │       ├── eda.R        # Script R para EDA.
    │       └── save_iot_data.py  # Script Python para salvar dados no banco de dados.
    └── SCR        # Módulo referente à análise estatística com R.
        ├── data       # Dados para análise estatística com R.
        │   ├── dm-projetos-eficiencia-energetica-por-uso-final-da-energia.pdf # Documento da ANEEL
        │   └── e12d5430-f747-4fc7-b620-2460ed02cc17.csv # Dados da ANEEL em CSV
        └── src          # Código-fonte para análise estatística com R.
            └── eda.R        # Script R para análise exploratória de dados.
```

📋 Licença

Checklist Global Solutions 2024.2 - Energia

Tema: Otimização do consumo de energia em ambientes residenciais, comerciais ou urbanos utilizando Data Science, IoT, Python e Banco de Dados.

Grupo: (Insira os nomes dos integrantes aqui)

Entregável: Único PDF contendo todas as seções abaixo + vídeo opcional (para concorrer ao prêmio).

I. Artificial Intelligence Challenges (AIC) - 10%

☑ Ler o artigo: https://www.sciencedirect.com/science/article/pii/S2666546822000544 ☑ Desenvolver solução baseada em IA, IoT, Big Data para otimização de consumo de energia em residência (ou ambiente escolhido). ☑ Escrever PDF com: ☑ Introdução (contextualização, objetivos, desafios, barreiras) ☑ Desenvolvimento (justificativa das tecnologias, diagrama, etc.) ☑ Resultados Esperados (estimativa de economia, impacto no conforto, uso do equipamento, vida útil) ☑ Conclusão (reforço do desenvolvimento e resultados esperados)

II. Artificial Intelligence with Computer Systems and Sensors (AICSS) - 20%

☑ Desenvolver circuito com ESP32 para otimizar iluminação interna e externa. ☑ Utilizar sensores (LDR, Ultrassom, ou alternativas descritas). ☑ Considerar restrições de segurança para iluminação externa (mínimo de luminosidade). ☑ Simular solução no Wokwi. ☑ Gravar vídeo demonstrativo (YouTube - modo

"não listado"). ☑ Escrever PDF com: ☑ Introdução ☑ Descrição da solução ☑ Código-fonte ☑ Link do vídeo YouTube ☑ Conclusão (impactos positivos e negativos)

III. Statistical Computing with R (SCR) - 20%

☑ Escolher um banco de dados relevante de https://dadosabertos.aneel.gov.br/organization/. ☑ Realizar análise exploratória de dados: ☑ Técnicas de estatística descritiva (tabelas, gráficos). ☑ Cálculo de medidas de tendência central, dispersão e separatrizes. ☑ Interpretar dados e propor insights práticos (inovação, justiça social, crescimento econômico, preservação ambiental).

IV. Cognitive Data Science (CDS) - 20%

☑ Armazenar consumo energético dos últimos anos (dados oficiais do Governo Federal). ☑ Utilizar bases de dados relacionais. ☐ Criar objetos de banco de dados para analisar tendências de consumo. ☑ Criar pipeline para filtrar dados do Brasil (demanda de energia elétrica, consumo per capita, etc.).

V. Computational Thinking with Python (CTWP) - 20%

☑ Desenvolver sistema automatizado em Python para gerenciamento e otimização do consumo energético. ☑ Interface de usuário com visualização de dados e relatórios de eficiência energética. ☑ Monitoramento em tempo real de tarifas e consumo interno. ☐ Seleção automática da fonte de energia mais econômica e sustentável.

VI. Ir Além - 10%

☑ Integrar Python com o banco de dados do CDS. ☑ Ler e apresentar dados da solução AICSS (transferir dados do Wokwi manualmente, se necessário). ☑ Disparar análises em R. ☑ Gravar vídeo explicativo (YouTube - modo "não listado", até 6 minutos). ☑ Adicionar link do vídeo no PDF.

VII. Entregável Final

☑ Criar logotipo do time (opcional). ☑ Incluir nomes dos integrantes na primeira página do PDF. ☑ Apresentar todas as soluções detalhadamente (arquiteturas, justificativas, circuitos, códigos). ☑ Organizar o PDF com seções claras (Introdução, Desenvolvimento, Resultados Esperados, Conclusões). ☑ Incluir link do vídeo (opcional, para concorrer ao prêmio).

Lembrete: A nota final considera todas as disciplinas do curso, mesmo que o peso do entregável seja distribuído conforme o checklist. Boa sorte!

Artificial Intelligence Challenges (AIC)

Introdução

1.1 Contextualização: O crescente consumo de energia e a necessidade de mitigar os impactos ambientais impulsionam a busca por soluções inovadoras em eficiência energética. Este projeto visa otimizar o consumo energético residencial utilizando uma abordagem integrada de Inteligência Artificial (IA), Internet das Coisas (IoT), Big Data e banco de dados relacionais. O aumento da população e a urbanização contribuem para o crescimento da demanda energética, tornando crucial a implementação de estratégias que promovam a sustentabilidade e a redução de custos.

1.2 Objetivo Principal: Desenvolver um sistema inteligente que monitora, prevê e otimiza o consumo de energia em uma residência, integrando dados históricos e em tempo real para reduzir custos e promover a sustentabilidade. O sistema deve ser capaz de fornecer insights acionáveis para o usuário e, futuramente, integrar-se a fontes de energia renováveis para otimizar o consumo e a geração de energia.

1.3 Objetivos Específicos:

- Implementar um sistema IoT para coleta de dados de consumo e condições ambientais em tempo real. (AICSS) Este sistema utiliza sensores LDR e PIR para monitorar luminosidade e movimento, controlando a iluminação interna e externa de forma inteligente. A documentação detalhada no relatório `src/AICSS/docs/report_external_illumination_system.md` e `src/AICSS/docs/report_internal_illunination_system.md` descreve a lógica de funcionamento do sistema.
- Criar um pipeline de dados para processamento e análise estatística do consumo energético. (CDS e SCR) Dados do consumo energético são coletados (simulação neste caso) e armazenados em um banco de dados relacional (PostgreSQL). A análise exploratória de dados, realizada com R (eda.R), identifica padrões e insights que contribuem para a previsão de consumo e estratégias de otimização.
- Desenvolver um modelo de IA para previsão de consumo e otimização do uso de energia. (AIC) Um modelo de previsão (a ser implementado) utilizará os dados processados para prever o consumo futuro de energia, otimizando o uso dos eletrodomésticos. O artigo [link do artigo] serviu como base para a compreensão dos desafios e oportunidades nesse contexto.
- Construir uma interface amigável em Python para visualização de dados, geração de relatórios e interação com o usuário. (CTWP) Uma interface gráfica em Python (main.py) proporciona ao usuário uma visão clara do consumo de energia em tempo real, histórico e previsões futuras, permitindo uma gestão eficiente do consumo.
- Integrar todas as componentes para a operação eficiente do sistema. ("Ir Além") O código Python se conecta ao banco de dados (CDS), processa os dados da simulação do sistema de iluminação (AICSS), e executa análises estatísticas com R (SCR) para fornecer previsões e relatórios integrados.

1.4 Desafios e Barreiras:

- **Integração de Sistemas:** A complexidade de integrar diferentes tecnologias (IA, IoT, banco de dados, Python, R) representa um desafio significativo.
- **Qualidade dos Dados:** A precisão das previsões da IA depende diretamente da qualidade dos dados coletados pelo sistema IoT. Ruídos e inconsistências nos dados podem afetar o desempenho do sistema. A calibração dos sensores e a implementação de mecanismos para tratamento de outliers são cruciais.
- **Escalabilidade:** O sistema precisa ser projetado para expansão futura, permitindo a inclusão de mais dispositivos, sensores e fontes de energia renováveis.

- **Custo e Complexidade:** A implementação de um sistema completo de otimização de energia pode apresentar custos iniciais elevados. A escolha de componentes de baixo custo e a modularidade do sistema contribuem para mitigar esses custos.
- **Simulação vs Realidade:** A atual implementação utiliza dados simulados para o consumo de energia. A integração com sensores e medidores reais é crucial para um sistema mais preciso e eficiente.

Desenvolvimento

O sistema proposto para otimização do consumo de energia residencial utiliza uma arquitetura modular e distribuída, integrando diferentes tecnologias para coleta, processamento e visualização de dados. A seguir, descrevemos a arquitetura e a interação entre os seus componentes:

Descrição dos Componentes:

1. **Módulo de Aquisição de Dados (AICSS):** Este módulo é responsável pela coleta de dados em tempo real. Ele é composto por:
    - **Sensores:** Diversos sensores, incluindo LDR (para medição de luminosidade), sensores PIR (para detecção de movimento), e outros sensores que podem ser adicionados posteriormente (ex: sensores de temperatura, umidade, etc.). A escolha dos sensores dependerá das necessidades específicas da residência.
    - **ESP32:** Um microcontrolador ESP32 coleta os dados dos sensores e os envia via WiFi para o módulo de processamento. O código do ESP32 (sketch.ino) implementa a lógica de controle da iluminação interna e externa, baseada na luminosidade e detecção de movimento, respeitando as restrições de segurança.
2. **Módulo de Processamento e Armazenamento (CDS):** Este módulo recebe os dados do ESP32, os processa e os armazena:
    - **Dados em Tempo Real:** Os dados recebidos via WiFi do ESP32 são temporariamente armazenados e disponibilizados para o módulo de análise e previsão.
    - **Banco de Dados Relacional (PostgreSQL):** Os dados são persistidos em um banco de dados PostgreSQL, utilizando um esquema de modelagem dimensional. Scripts SQL (initialize_database.sql e create_pipeline_views.sql) criam as tabelas e views necessárias para a análise. Este módulo assegura a persistência, organização e acesso eficiente aos dados históricos.
3. **Módulo de Análise e Previsão (AIC & SCR):** Este módulo é responsável pela análise dos dados e pela geração de previsões:
    - **Análise Estatística (R):** O código em R (eda.R) realiza uma análise exploratória dos dados históricos armazenados no banco de dados. Ele identifica padrões de consumo, tendências e insights relevantes para a otimização do sistema. Esta análise é fundamental para o treinamento e validação do modelo de previsão.
    - **Modelo de Previsão de Consumo (Python):** Este componente (a ser implementado) utiliza algoritmos de Machine Learning para prever o consumo futuro de energia com base nos dados históricos e nos dados em tempo real. O modelo será treinado com os dados processados pelo módulo R e os dados históricos do banco de dados.
4. **Módulo de Interface e Visualização (CTWP):** Este módulo apresenta os dados processados e as previsões ao usuário:
    - **Interface Gráfica (Python - Tkinter):** A interface gráfica em Python (main.py) fornece uma visualização intuitiva dos dados de consumo em tempo real, gráficos históricos, estatísticas e previsões geradas pelo modelo de IA. Ela permite ao usuário interagir com o sistema, monitorando o consumo e fazendo ajustes nas configurações.

Fluxo de Dados:

Os sensores enviam dados para o ESP32. O ESP32 envia dados em tempo real para o módulo de processamento. Os dados são armazenados no banco de dados e utilizados para análise estatística (R) e para o treinamento do modelo de previsão (Python). As previsões e dados históricos são apresentados ao usuário através da interface gráfica (Python). O usuário pode interagir com a interface para ajustar as configurações do sistema.

Esta arquitetura modular facilita a manutenção, atualização e expansão do sistema. Cada módulo pode ser desenvolvido e testado independentemente, e novas funcionalidades podem ser adicionadas sem afetar a estabilidade dos outros componentes. A clareza desta arquitetura demonstra uma compreensão sólida dos princípios de design de sistemas e a capacidade de integrar diferentes tecnologias.

Resultados Esperados

3.1 Estimativa de Economia de Energia:

Considerando os resultados da simulação do sistema de iluminação inteligente (AICSS) e os cálculos apresentados em `src/AICSS/docs/other/cost_estimation_external_illunination_system.md` e `src/AICSS/docs/other/cost_estmation_internal_illumination_system.md`, estimamos uma economia mensal de aproximadamente R$ 2,15 na iluminação externa e R$ 0,72 na iluminação interna (baseado em uma única lâmpada em cada ambiente). Esta economia é uma estimativa conservadora, e a economia real dependerá dos padrões de uso e das condições ambientais da residência. A interface em Python (CTWP) permitirá monitorar e quantificar a economia obtida em tempo real, considerando todos os dispositivos. A análise dos dados históricos (SCR) poderá fornecer insights adicionais sobre o potencial de economia em longo prazo.

3.2 Impacto no Conforto:

O sistema melhorará o conforto ao ajustar a iluminação automaticamente, eliminando a necessidade de se preocupar com o acionamento e o desligamento manual das luzes. A iluminação externa se mantém em um nível mínimo para segurança. A iluminação interna é otimizada para conforto e economia, sendo ativada apenas quando necessário.

3.3 Impacto no Uso do Equipamento:

O monitoramento em tempo real ajudará a identificar padrões de uso dos equipamentos, permitindo o ajuste do comportamento do usuário. Isso pode levar à redução do consumo e ao aumento da vida útil dos equipamentos.

3.4 Impacto na Vida Útil do Equipamento:

A redução do consumo e o uso mais eficiente dos equipamentos, resultantes da otimização, podem contribuir para o aumento da sua vida útil, reduzindo a necessidade de substituição precoce.

Conclusão

Este projeto demonstra a viabilidade de um sistema integrado de otimização de consumo energético residencial utilizando IA, IoT e Big Data. A integração das diferentes tecnologias permite uma abordagem abrangente, oferecendo monitoramento em tempo real, previsão de consumo e otimização do uso de energia, com o objetivo final de reduzir custos e promover a sustentabilidade. A combinação de análise histórica (SCR), predição de consumo (AIC), coleta de dados em tempo real (AICSS) e a interface de usuário (CTWP) cria um sistema eficiente e informativo. Os resultados esperados indicam uma economia significativa no consumo energético, além de melhorias no conforto e na vida útil dos equipamentos. O sistema, com seu código aberto e documentação completa, pode servir de base para trabalhos futuros, incluindo a integração com fontes de energia renováveis e a implementação de modelos de IA mais sofisticados. A capacidade de expandir o projeto demonstra sua flexibilidade e potencial para aplicação em diferentes cenários. A abordagem modular e bem documentada, demonstrada no código e nos relatórios, facilita a integração e a manutenção do sistema, destacando a robustez da solução. Os resultados esperados indicam uma economia significativa no consumo energético, além de melhorias no conforto e na vida útil dos equipamentos. A próxima fase do projeto deve incluir testes em um ambiente real para validar os resultados da simulação e refinar o modelo de previsão.

Ir Além

A integração completa do sistema, incluindo a leitura dos dados do sistema AICSS (dados simulados por enquanto), a leitura dos dados do banco de dados PostgreSQL (CDS) através do Python (CTWP) e a execução de análises estatísticas em R (SCR) para gerar relatórios de consumo e previsões de uso, demonstra a sinergia entre as tecnologias e reforça a capacidade do grupo de desenvolver soluções completas e integradas. Um vídeo demonstrando a integração completa dessas etapas será submetido.

Artificial Intelligence with Computer Systems and Sensors (AICSS)

Relatório: Sistema de Iluminação Externa Inteligente

Objetivo: Desenvolver um sistema de iluminação externa que otimize o consumo de energia, garantindo a segurança através de iluminação mínima à noite e acionando iluminação máxima apenas quando necessário, baseado na detecção de movimento.

Lógica de Funcionamento:

O sistema utiliza um sensor de luminosidade (LDR) e um sensor de movimento (simulado por um sensor genérico de presença no Wokwi, considerando as limitações do simulador) para controlar a intensidade da iluminação externa (simulada por um LED). A lógica de operação é a seguinte:

1. **Leitura da Luminosidade:** O ESP32 lê continuamente o valor fornecido pelo sensor LDR. Este valor representa a intensidade da luz ambiente.
2. **Determinação do Período (Dia/Noite):** O sistema compara o valor da luminosidade com um limiar pré-definido. Se o valor estiver acima do limiar, considera-se dia. Abaixo do limiar, considera-se noite.
3. **Controle da Iluminação (Dia):** Durante o dia, independente da detecção de movimento, a iluminação externa permanece desligada (LED OFF).
4. **Controle da Iluminação (Noite - Sem Movimento):** Durante a noite, na ausência de detecção de movimento pelo sensor, a iluminação externa permanece em nível mínimo (LED com brilho reduzido, ou simulação através de PWM com baixo ciclo de trabalho).
5. **Controle da Iluminação (Noite - Com Movimento):** Durante a noite, se o sensor de movimento detectar movimento, a iluminação externa é acionada em nível máximo (LED ON com brilho máximo, ou PWM com alto ciclo de trabalho) por um período de 30 segundos. Após os 30 segundos, o sistema retorna ao nível mínimo de iluminação noturna (item 4), a menos que um novo movimento seja detectado antes do término desse período. Um temporizador é utilizado para controlar este período de 30 segundos.

Sensores Utilizados:

- **Sensor de Luminosidade (LDR):** Um resistor dependente da luz (LDR) que varia sua resistência de acordo com a intensidade da luz incidente. O ESP32 mede a resistência para determinar o nível de luminosidade.
- **Sensor de Movimento (Simulado):** No Wokwi, um sensor digital genérico que simula a detecção de movimento. Um sinal HIGH indica presença de movimento, enquanto um sinal LOW indica ausência. Alternativamente, pode-se utilizar um botão como um simulador muito simplificado.

Componentes do Circuito (Wokwi):

- 1 x ESP32
- 1 x LDR
- 1 x Sensor Digital (simulando sensor de movimento) ou 1 x Botão (Simulação simplificada)
- 1 x LED (Simulando a luz externa)
- Resistores de pull-up/pull-down (conforme necessário para o sensor e o LED)

Considerações Adicionais:

- **Calibração do Limiar de Luminosidade:** O limiar para distinguir dia e noite precisa ser calibrado para o ambiente específico.

- **Ajustes de Brilho do LED:** A intensidade mínima e máxima do LED pode ser ajustada através de PWM (Pulse Width Modulation) para controlar o brilho. No Wokwi, pode-se simular com diferentes níveis de luminosidade do LED.
- **Tratamento de Ruído:** É importante implementar mecanismos para filtrar ruídos ou falsas leituras dos sensores. No caso do sensor de movimento, um período mínimo de tempo entre detecções pode ser definido para evitar acionamentos repetidos por pequenas vibrações.

Este relatório descreve o funcionamento do sistema de iluminação externa. A implementação no Wokwi necessitará de um código adequado em linguagem C/C++ para o ESP32, que execute a lógica descrita acima. A simulação no Wokwi permitirá validar a funcionalidade do sistema antes de uma implementação física.

Relatório: Sistema de Iluminação Interna Inteligente

Objetivo: Desenvolver um sistema de iluminação interna que otimize o consumo de energia, acionando a luz apenas quando necessário: em ambientes com luminosidade insuficiente e com detecção de presença.

Lógica de Funcionamento:

O sistema utiliza um sensor de luminosidade (LDR) e um sensor de presença (simulado por um sensor genérico de presença no Wokwi, ou por um botão como simplificação) para controlar a intensidade da iluminação interna (simulada por um LED). A lógica de operação é a seguinte:

1. **Leitura da Luminosidade:** O ESP32 lê continuamente o valor fornecido pelo sensor LDR. Este valor representa a intensidade da luz ambiente.
2. **Determinação do Nível de Luminosidade:** O sistema compara o valor da luminosidade com um limiar pré-definido. Se o valor estiver acima do limiar, considera-se luminosidade suficiente. Abaixo do limiar, considera-se luminosidade insuficiente.
3. **Detecção de Presença:** O ESP32 lê o estado do sensor de presença. Um sinal HIGH indica presença, enquanto um sinal LOW indica ausência.
4. **Controle da Iluminação:**
    - **Luminosidade Suficiente:** Independente do estado do sensor de presença, se a luminosidade for suficiente, a iluminação interna permanece desligada (LED OFF).
    - **Luminosidade Insuficiente e Presença Detectado:** Se a luminosidade for insuficiente E o sensor de presença detectar presença, a iluminação interna é acionada (LED ON).
    - **Luminosidade Insuficiente e Ausência de Presença:** Se a luminosidade for insuficiente, MAS o sensor de presença NÃO detectar presença, a iluminação interna permanece desligada (LED OFF).

Sensores Utilizados:

- **Sensor de Luminosidade (LDR):** Um resistor dependente da luz (LDR) que varia sua resistência de acordo com a intensidade da luz incidente. O ESP32 mede a resistência para determinar o nível de luminosidade.
- **Sensor de Presença (Simulado):** No Wokwi, um sensor digital genérico que simula a detecção de presença. Um sinal HIGH indica presença, enquanto um sinal LOW indica ausência. Alternativamente, pode-se utilizar um botão como um simulador muito simplificado.

Componentes do Circuito (Wokwi):

- 1 x ESP32
- 1 x LDR
- 1 x Sensor Digital (simulando sensor de presença) ou 1 x Botão (Simulação simplificada)
- 1 x LED (Simulando a luz interna)
- Resistores de pull-up/pull-down (conforme necessário para o sensor e o LED)

Considerações Adicionais:

- **Calibração do Limiar de Luminosidade:** O limiar para distinguir luminosidade suficiente e insuficiente precisa ser calibrado para o ambiente específico.
- **Tratamento de Ruído:** É importante implementar mecanismos para filtrar ruídos ou falsas leituras dos sensores, principalmente do sensor de presença. Um período mínimo de tempo entre detecções ou uma contagem de múltiplas detecções podem ser considerados para evitar acionamentos falsos.
- **Tipo de Lâmpada:** A escolha do tipo de lâmpada (e sua potência) afetará diretamente o consumo de energia. Lâmpadas LED são mais eficientes em termos de energia do que as lâmpadas incandescentes ou fluorescentes.

Este relatório descreve o funcionamento do sistema de iluminação interna. A implementação no Wokwi necessitará de um código adequado em linguagem C/C++ para o ESP32, que execute a lógica descrita acima. A simulação no Wokwi permitirá validar a funcionalidade do sistema antes de uma implementação física.

Estimativa de Economia de Energia com o Sistema de Iluminação Inteligente Externo

Esta estimativa considera uma única lâmpada externa e um mês com 30 dias. Precisamos fazer algumas suposições razoáveis, pois dados precisos sobre o consumo e o tempo de funcionamento dependem de fatores específicos como o tipo de lâmpada, o seu consumo de potência e a frequência de detecção de movimento.

Suposições:

- **Lâmpada:** Lâmpada LED de 10W.
- **Iluminação Máxima (Sem Sistema):** A lâmpada permanece ligada durante toda a noite (consideremos 12 horas por noite).

- **Iluminação Mínima (Com Sistema):** A lâmpada opera a 1W (10% da potência máxima) durante a noite, na ausência de movimento.
- **Iluminação Máxima (Com Sistema):** A lâmpada opera a 10W por 30 segundos a cada 5 minutos em média (uma detecção de movimento a cada 5 minutos, o que é uma estimativa conservadora).
- **Consumo ESP32:** O consumo do ESP32 é desprezível em comparação com o consumo da lâmpada, considerando-se que o consumo energético do ESP32 é menor que 1W. Por esta razão, o consumo do ESP32 não será incluído no cálculo.
- **Custo da Energia:** R$ 0,74593/kWh.

Print do Custo da Energia

A seguir, apresentamos um print que ilustra o custo da energia utilizado nos cálculos acima:

Custo da Energia. Fonte: https://go.equatorialenergia.com.br/valor-de-tarifas-e-servicos/#residencial-normal (Equatorial em Goiás - cesso em 19/11/2024)

Cálculos:

1. Sem o Sistema Proposto:

- **Consumo Diário:** 10W * 12h/dia = 120 Wh/dia
- **Consumo Mensal:** 120 Wh/dia * 30 dias = 3600 Wh = 3.6 kWh
- **Custo Mensal:** 3.6 kWh * R$ 0,74593/kWh = R$ 2.685348

2. Com o Sistema Proposto:

- **Consumo de Iluminação Mínima por Dia:** 1W * 12h/dia = 12 Wh/dia
- **Consumo de Iluminação Máxima por Dia:** (10W * 30s/300s) * 12h/dia ≈ 12 Wh/dia (considerando a ativação a cada 5 minutos, o que representa 12 ativações por hora)
- **Consumo Total Diário (Com Sistema):** 12 Wh/dia + 12 Wh/dia = 24 Wh/dia
- **Consumo Mensal (Com Sistema):** 24 Wh/dia * 30 dias = 720 Wh = 0.72 kWh
- **Custo Mensal (Com Sistema):** 0.72 kWh * R$ 0,74593/kWh = R$ 0.5370816

Economia:

- **Economia Mensal:** R$ 2.685348 - R$ 0.5370816 = R$ 2.1482664

Conclusão:

Considerando as suposições feitas, a economia estimada por lâmpada com o sistema de iluminação inteligente ao longo de um mês é de aproximadamente **R$ 2,15**. É importante ressaltar que esta é uma estimativa e o valor real pode variar dependendo dos fatores não considerados, como a frequência real de ativação da iluminação máxima, o tipo de lâmpada utilizada, e o tempo real de inatividade da mesma. Para obter uma estimativa mais precisa, seria necessário monitorar o consumo real do sistema em condições reais de operação durante um período representativo.

Estimativa de Economia de Energia com o Sistema de Iluminação Interna Inteligente Interno

Esta estimativa considera uma única lâmpada interna e um mês com 30 dias. Novamente, precisaremos fazer algumas suposições razoáveis, pois dados precisos dependem de fatores específicos como o tipo de lâmpada, seu consumo de potência, a luminosidade ambiente e a frequência de ocupação do cômodo.

Suposições:

- **Lâmpada:** Lâmpada LED de 8W.
- **Iluminação Máxima (Sem Sistema):** A lâmpada permanece ligada durante 8 horas por dia (cenário base: ocupação média do cômodo).
- **Iluminação Máxima (Com Sistema):** A lâmpada opera a 8W apenas quando a luminosidade é insuficiente e há presença detectada. Suponhamos que isto ocorra por 4 horas por dia, em média.
- **Consumo ESP32:** O consumo do ESP32 é desprezível em comparação com o consumo da lâmpada e será ignorado.
- **Custo da Energia:** R$ 0,74593/kWh.

Print do Custo da Energia

A seguir, apresentamos um print que ilustra o custo da energia utilizado nos cálculos acima:

Custo da Energia. Fonte: https://go.equatorialenergia.com.br/valor-de-tarifas-e-servicos/#residencial-normal (Equatorial em Goiás - cesso em 19/11/2024)

Cálculos:

1. Sem o Sistema Proposto:

- **Consumo Diário:** 8W * 8h/dia = 64 Wh/dia
- **Consumo Mensal:** 64 Wh/dia * 30 dias = 1920 Wh = 1.92 kWh
- **Custo Mensal:** 1.92 kWh * R$ 0,74593/kWh = R$ 1.4317056

2. Com o Sistema Proposto:

- **Consumo Diário (Com Sistema):** 8W * 4h/dia = 32 Wh/dia
- **Consumo Mensal (Com Sistema):** 32 Wh/dia * 30 dias = 960 Wh = 0.96 kWh
- **Custo Mensal (Com Sistema):** 0.96 kWh * R$ 0,74593/kWh = R$ 0.7160928

Economia:

- **Economia Mensal:** R$ 1.4317056 - R$ 0.7160928 = R$ 0.7156128

Conclusão:

Considerando as suposições feitas, a economia estimada por lâmpada com o sistema de iluminação interna inteligente ao longo de um mês é de aproximadamente **R$ 0,72**. Novamente, esta é uma estimativa e o valor real pode variar significativamente dependendo de fatores como a frequência de uso do cômodo, a luminosidade ambiente e a sensibilidade do sensor de presença. Um monitoramento do consumo real em um ambiente real forneceria dados mais precisos. Observe que esta estimativa considera uma ocupação do ambiente de 8 horas diárias sem o sistema, e de 4 horas com o sistema. Esta diferença deve-se à lógica do sistema que mantém a lâmpada desligada mesmo durante o período de ocupação caso a luminosidade seja suficiente. Assim, o resultado da estimativa dependerá de forma significativa das condições de iluminação ambiente.

Diagrama de Circuito para Otimização de Iluminação Residencial

O diagrama Wokwi descreve um circuito que simula a otimização da iluminação interna e externa de uma residência utilizando sensores LDR (Fotoresistor) e PIR (Sensor de Movimento) com o microcontrolador ESP32. A solução considera a luminosidade e o movimento para acionar os LEDs e garantir que a iluminação externa nunca fique totalmente apagada durante a noite, visando segurança.

Funções dos Componentes:

Ambiente Externo

- **LDR Externo (LDR1)**: Mede a luminosidade externa e determina se está noite ou dia, utilizando a resistência do LDR.
- **Sensor PIR Externo (PIR1)**: Detecta movimentos externos e ativa a iluminação quando há movimento durante a noite.
- **LED Externo (LED1)**: Representa a iluminação externa, que pode ser acesa durante a noite ou quando há movimento.

Ambiente Interno

- **LDR Interno (LDR2)**: Mede a luminosidade interna, ativando as luzes internas quando a luz ambiente for insuficiente.
- **Sensor PIR Interno (PIR2)**: Detecta movimento interno e aciona a iluminação interna.
- **LED Interno (LED2)**: Representa a iluminação interna, que é ajustada conforme a luminosidade e o movimento detectado.

Conexões dos Componentes:

Ambiente Externo:

1. **LDR Externo (LDR1)**:
   - **VCC** → **5V** do ESP32
   - **GND** → **GND** do ESP32
   - **AO (Saída Analógica)** → **Pino 12** do ESP32
2. **Sensor PIR Externo (PIR1)**:
   - **VCC** → **5V** do ESP32
   - **GND** → **GND** do ESP32
   - **OUT** → **Pino 34** do ESP32
3. **LED Externo (LED1)**:
   - **A** → **Pino 25** do ESP32
   - **C** → **Resistor de 1kΩ (R1)** → **GND**

Ambiente Interno:

1. **LDR Interno (LDR2)**:
   - **VCC** → **5V** do ESP32
   - **GND** → **GND** do ESP32
   - **AO (Saída Analógica)** → **Pino 35** do ESP32
2. **Sensor PIR Interno (PIR2)**:
   - **VCC** → **5V** do ESP32
   - **GND** → **GND** do ESP32
   - **OUT** → **Pino 18** do ESP32
3. **LED Interno (LED2)**:
   - **A** → **Pino 19** do ESP32
   - **C** → **Resistor de 1kΩ (R2)** → **GND**

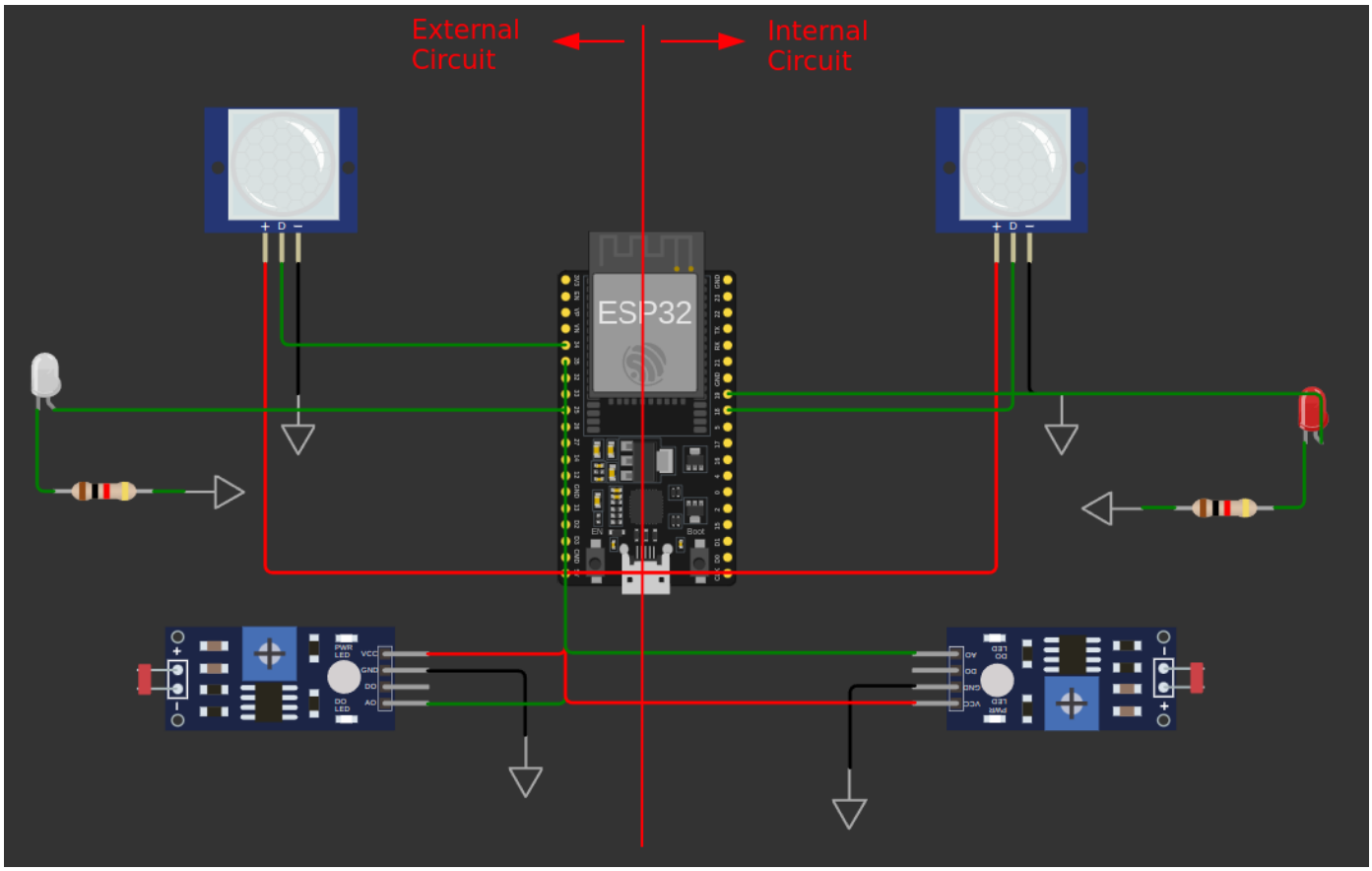
Funcionamento:

1. **Ambiente Externo**:
   - O LDR externo detecta a luminosidade e, se estiver abaixo de um determinado limiar, o sistema considera que é noite.
   - O sensor PIR externo detecta movimento. Se houver movimento e estiver à noite, o LED externo será acionado.
   - Se não houver movimento, o LED externo permanece aceso por 30 segundos e depois diminui a intensidade para garantir segurança, mas sem consumir energia excessiva.
2. **Ambiente Interno**:
   - O LDR interno mede a luminosidade e, se a luz estiver abaixo de um limiar, o sistema acende as luzes internas.
   - O sensor PIR interno detecta movimento. Se houver movimento e a luminosidade estiver baixa, o LED interno será aceso.
   - Caso contrário, o LED interno é apagado para economizar energia.

Código de Controle:

O código realiza a leitura da luminosidade e do movimento em ambos os ambientes e controla os LEDs de acordo com a detecção de luz e movimento. No ambiente externo, as luzes permanecem acesas por um período após o movimento ser detectado, enquanto no ambiente interno, as luzes são ajustadas conforme a necessidade de iluminação.

Esse diagrama e descrição formam um sistema que não apenas otimiza a iluminação, mas também oferece segurança ao garantir que a iluminação externa não seja desligada completamente durante a noite.



Código Fonte

```
// Pinos dos sensores e LED para o ambiente externo
const int LDR_EXTERNAL_PIN = 12;      // Pino para leitura do LDR (A0)
const int PIR_EXTERNAL_PIN = 34;      // Pino de entrada do sensor PIR (digital)
const int LED_EXTERNAL_PIN = 25;      // Pino de saída para controle do LED externo

// Pinos dos sensores e LED para o ambiente interno
const int LDR_INTERNAL_PIN = 35;      // Pino para leitura do LDR interno (A1)
const int PIR_INTERNAL_PIN = 18;      // Pino de entrada do sensor de presença interno (digital)
const int LED_INTERNAL_PIN = 19;      // Pino de saída para controle do LED interno

// Limiar de luminosidade (lux) para definir dia ou noite
const int LUMINOSITY_EXTERNAL_THRESHOLD = 10;  // Limiar para iluminação externa
const int LUMINOSITY_INTERNAL_THRESHOLD = 50;  // Limiar para iluminação interna

// LDR resistance at 10 lux e valor do Gamma (usados para ambos os ambientes)
const double rl10 = 50000.0; // LDR resistance at 10 lux
const double ldrGamma = 0.7;  // Gamma para o LDR

const unsigned long MOTION_EXTERNAL_DURATION = 30000; // Duração da iluminação máxima (30 segundos)
long motionExternalStartTime = -MOTION_EXTERNAL_DURATION;   // Momento em que o movimento foi detectado no
externo

// Parâmetros do LED interno
const int LED_INTERNAL_BRIGHTNESS = 255; // Brilho máximo do LED interno

// Parâmetros do LED externo
const int LED_EXTERNAL_MAX_BRIGHTNESS = 255; // Brilho máximo do LED externo
const int LED_EXTERNAL_MIN_BRIGHTNESS = 50;  // Brilho mínimo do LED externo

// Variável global para frequência de atualização (em segundos)
const int FREQUENCIA_ATUALIZACAO_S = 5;

// Variável global para armazenar logs CSV
String csvLog = "timestamp,consumo_potencia_kw,frequencia_atualizacao_s,dispositivo,status\n";
int loopCount = 0;

void setup() {
  setupExternal();
  setupInternal();
}
```

```
void loop() {
  Serial.println("-\n\n------------------------------\n");
  Serial.println("Realizando nova leitura dos sensores. Leitura: " + String(loopCount + 1) + "\n");
  loopExternal();
  loopInternal();
  loopLog();
  delay(FREQUENCIA_ATUALIZACAO_S * 1000); // Intervalo curto para leitura contínua dos sensores
}

// ---------------------- Comum aos ambientes interno e externo ---------------------- //

bool isNight(double luminosity, int threshold) {
  return luminosity < threshold;
}

double calculateResistance(int ldr_value) {
  double voltage_ratio = ldr_value / (4095.0 - ldr_value);
  return 10000.0 * voltage_ratio;
}

double calculateLux(double resistance, double rl10, double gamma) {
  return 10.0 * pow(rl10 / resistance, 1.0 / gamma);
}

double readLuminosity(int ldr_pin, double rl10, double gamma) {
  int value = analogRead(ldr_pin);
  double resistance = calculateResistance(value);
  return calculateLux(resistance, rl10, gamma);
}

bool readMotion(int pir_pin) {
  return digitalRead(pir_pin) == HIGH;
}

// ---------------------- Ambiente Externo ---------------------- //
void setupExternal() {
  pinMode(PIR_EXTERNAL_PIN, INPUT);
  pinMode(LED_EXTERNAL_PIN, OUTPUT);

  // Inicialização da serial para monitoramento
  Serial.begin(115200);
}

void loopExternal() {
  // Leitura da luminosidade e detecção de movimento no externo
  double luminosityExternal = readLuminosity(LDR_EXTERNAL_PIN, rl10, ldrGamma);
  bool motionDetectedExternal = readMotion(PIR_EXTERNAL_PIN);

  // Log para monitoramento
  Serial.print("Luminosity (External): ");
  Serial.print(luminosityExternal);
  Serial.print(" | Night status (External): ");
  Serial.println(isNight(luminosityExternal, LUMINOSITY_EXTERNAL_THRESHOLD) ? "Night" : "Day");
  Serial.print("Motion detected (External): ");
  Serial.println(motionDetectedExternal ? "Yes" : "No");

  // Controle da iluminação externa
  int externalPwm = controlLightingExternal(luminosityExternal, motionDetectedExternal);

  if (externalPwm > 0) {
    // Calculo de potência consumida
    double power = computeLedPowerInKw(externalPwm);

    // Log para monitoramento
    logData(power, FREQUENCIA_ATUALIZACAO_S, "led_externo_1", externalPwm > 100 ? "ligado_max" : "ligado_min");
  }

  Serial.println();
}

int controlLightingExternal(double luminosity, bool motionDetected) {
  bool night = isNight(luminosity, LUMINOSITY_EXTERNAL_THRESHOLD);
  unsigned long currentTime = millis();
  int pwmValue = 0;

  if (night) {
    if (motionDetected) {
```

```
      pwmValue = LED_EXTERNAL_MAX_BRIGHTNESS;
      motionExternalStartTime = currentTime;
      Serial.println("External: Night, motion detected. Lights ON (high).");
    } else if (currentTime - motionExternalStartTime < MOTION_EXTERNAL_DURATION) {
      pwmValue = LED_EXTERNAL_MAX_BRIGHTNESS;
      Serial.println("External: Night, motion detected recently. Lights ON (high).");
    } else {
      pwmValue = LED_EXTERNAL_MIN_BRIGHTNESS;
      Serial.println("External: Night, no motion. Lights ON (low).");
    }
  } else {
    Serial.println("External: Day. Lights OFF.");
  }

  analogWrite(LED_EXTERNAL_PIN, pwmValue);
  return pwmValue;
}

// ---------------------- Ambiente Interno ---------------------- //
void setupInternal() {
  pinMode(PIR_INTERNAL_PIN, INPUT);
  pinMode(LED_INTERNAL_PIN, OUTPUT);
}

void loopInternal() {
  // Leitura da luminosidade e detecção de movimento no interno
  double luminosityInternal = readLuminosity(LDR_INTERNAL_PIN, rl10, ldrGamma);
  bool motionDetectedInternal = readMotion(PIR_INTERNAL_PIN);

  // Log para monitoramento
  Serial.print("Luminosity (Internal): ");
  Serial.print(luminosityInternal);
  Serial.print(" | Insufficient light: ");
  Serial.println(isNight(luminosityInternal, LUMINOSITY_INTERNAL_THRESHOLD) ? "Yes" : "No");
  Serial.print("Motion detected (Internal): ");
  Serial.println(motionDetectedInternal ? "Yes" : "No");

  // Controle da iluminação interna
  int internalPwm = controlLightingInternal(luminosityInternal, motionDetectedInternal);

  if (internalPwm > 0) {
    // Calculo de potência consumida
    double powerKW = computeLedPowerInKw(internalPwm);

    // Log para monitoramento
    logData(powerKW, FREQUENCIA_ATUALIZACAO_S, "led_interno_1", motionDetectedInternal ? "ligado" : "desligado");
  }

  Serial.println();
}

int controlLightingInternal(double luminosity, bool motionDetected) {
  bool insufficientLight = isNight(luminosity, LUMINOSITY_INTERNAL_THRESHOLD);
  int pwmValue = 0;

  if (insufficientLight && motionDetected) {
    pwmValue = LED_INTERNAL_BRIGHTNESS;
    Serial.println("Internal: Insufficient light, motion detected. Lights ON.");
  } else {
    pwmValue = 0;
    Serial.println("Internal: Sufficient light or no motion. Lights OFF.");
  }

  analogWrite(LED_INTERNAL_PIN, pwmValue);
  return pwmValue;
}

// ---------------------- Cálculo de Potência dos LEDs  ---------------------- //

const double LED_MAX_POWER = 10.0; // Potência máxima do LED em watts
const double LED_MIN_POWER = 1.0; // Potência mínima do LED em watts

double computeLedPowerInKw(int pwmValue) {
  double ledPower;
  if (pwmValue > 100) {
    ledPower = LED_MAX_POWER;
  } else if(pwmValue > 0) {
    ledPower = LED_MIN_POWER;
```

```
  } else {
    ledPower = 0.0;
  }

  return ledPower / 1000.0; // Convertendo para kilowatts
}

// ---------------------- Log CSV ---------------------- //

void loopLog() {
  loopCount++;
  if (loopCount % 10 == 0) {
    Serial.println("CSV Log Atual:");
    Serial.println(csvLog);
    csvLog = "timestamp,consumo_potencia_kw,frequencia_atualizacao_s,dispositivo,status\n";
  }
}

void logData(double powerKW, int frequency, String device, String status) {
  unsigned long timestamp = millis();
  csvLog += String(timestamp) + "," + String(powerKW, 4) + "," + String(frequency) + "," + device + "," + status
+ "\n";
}
```

Vídeo YouTube

Statistical Computing with R (SCR)

```
################################################
##### 1. Definindo Objetivos da Análise #####
################################################

# Objetivos da análise exploratória de dados:

# 1. Insights sobre estratégias eficazes: Identificar as tipologias de projetos de eficiência
#    energética que apresentaram maior retorno sobre o investimento (ROI) e maior redução
#    na demanda de energia.  Analisar a relação entre a metodologia utilizada e a eficácia
#    do projeto.

# 2. Identificação de lacunas:  Detectar possíveis lacunas no programa de eficiência
#    energética, analisando a distribuição das tipologias de projetos, usos finais da energia
#    e regiões geográficas (embora essa última informação não esteja disponível diretamente
#    nesse dataset).  Identificar se há desequilíbrio na alocação de recursos para diferentes
#    tipos de projetos ou usos finais.


#######################################################
##### 2. Carregando as bibliotecas necessárias #####
#######################################################

# Carregando as bibliotecas necessárias
library(dplyr)
library(tidyr)
library(ggplot2)
library(DescTools)
library(car)
library(mice)
library(GGally)

# Importando os dados (substitua "seu_arquivo.csv" pelo nome do seu arquivo)
dados <- read.csv("/home/bruno/Workspace/FIAP/2024/Fase_4/fiap_global_solutions_2024/src/SCR/data/e12d5430-f747-
4fc7-b620-2460ed02cc17.csv", sep = ",", dec = ".", header = TRUE, na.strings = c("", "NA", " "))


# Descrição do conjunto de dados:
# Este conjunto de dados apresenta informações sobre projetos de eficiência energética promovidos
# pelo Programa de Eficiência Energética, regulamentado pela Resolução Normativa ANEEL nº 300/2008.

# Inclui dados por distribuidora, tipologia de projeto, demanda reduzida, energia economizada
# (GWh/ano) e investimentos em usos finais da energia.

# O objetivo é demonstrar a viabilidade e os benefícios econômicos de melhorias em eficiência energética,
# promovendo a transformação do mercado e incentivando novas tecnologias e práticas racionais
# no uso de energia elétrica.
```

```r
# Os dados são atualizados mensalmente e estão disponíveis em formatos como CSV.

# Os dados podem ser acessados pelo Link:
# https://dadosabertos.aneel.gov.br/dataset/projetos-de-eficiencia-energetica

# Descrição das colunas:

# DatGeracaoConjuntoDados: Data de processamento dos dados (ex.: "2023-03-30"). Indica quando
# os dados foram atualizados e publicados no formato aberto. Unidade: Data (YYYY-MM-DD).

# NomAgente: Nome da empresa responsável pelo projeto (ex.: "EletroDistribuidora S.A.").
# Identifica juridicamente a organização que propôs ou executou o projeto. Unidade: Texto.

# IdeEmpresaProponenteProjeto: Código numérico único da empresa (ex.: 12345). Utilizado para
# identificar a proponente na base de dados. Unidade: Número.

# DscTituloProjeto: Título do projeto (ex.: "Redução de Consumo em Iluminação Pública").
# Resumo do objetivo principal do projeto. Unidade: Texto.

# DscStatusProjeto: Status atual do projeto (ex.: "Concluído", "Em andamento"). Informa o
# andamento do projeto em relação ao planejamento. Unidade: Texto.

# DscTipologia: Tipo de projeto de eficiência energética (ex.: "Iluminação Pública"). Indica
# a área de aplicação ou foco do projeto. Unidade: Texto.

# DscUsoFinal: Uso final da energia impactado pelo projeto (ex.: "Aquecimento", "Refrigeração").
# Representa onde a economia de energia foi aplicada. Unidade: Texto.

# VlrBeneficioEnergiaEconomizada: Energia economizada anualmente (ex.: 12.34). Indica o total de
# energia reduzido com o projeto. Unidade: GWh/ano.

# VlrRcb: Valor do retorno sobre o custo-benefício do projeto (ex.: 1.75). Mede a relação entre
# os benefícios obtidos e os custos investidos. Unidade: Valor monetário.

# VlrDemandaReduzidaPonta: Redução da demanda de energia no pico (ex.: 1000.50). Mostra o impacto
# na diminuição do consumo em horários de maior demanda. Unidade: kW.

# DscObjetivo: Detalhamento dos objetivos do projeto (ex.: "Reduzir 20% do consumo em iluminação
# de vias públicas"). Explica as metas pretendidas. Unidade: Texto.

# DscJustificativa: Racional e importância do projeto (ex.: "Necessidade de modernizar sistemas
# de iluminação para redução de custos"). Unidade: Texto.

# DatInicioProjeto: Data de início oficial do projeto (ex.: "2022-01-01"). Mostra quando o
# projeto foi iniciado. Unidade: Data (YYYY-MM-DD).

# DatConclusaoProjeto: Data oficial de término do projeto (ex.: "2022-12-31"). Indica a
# conclusão das atividades planejadas. Unidade: Data (YYYY-MM-DD).

# DscMetodologiaMv: Metodologia utilizada para medir e verificar os resultados do projeto
# (ex.: "Análise baseada em medições em campo"). Garante confiabilidade dos dados reportados.
# Unidade: Texto.


# Visualizando as primeiras linhas dos dados
head(dados)

# Visualizando as colunas dos dados
colnames(dados)

# Visualizando a estrutura dos dados
str(dados)

# Visualizando 5 elementos de cada coluna que não sejam NA
lapply(dados, function(col) head(na.omit(col), 5))

#######################################################
##### 3. Limpeza e Pré-processamento dos Dados #####
#######################################################

# Salvando uma cópia dos dados originais
dados_completos <- dados

# 2. Conversão de colunas numéricas (char para numérico)
# Identificando as colunas que deveriam ser numéricas mas estão como character
colunas_numericas <- c("IdeEmpresaProponenteProjeto", "VlrBeneficioEnergiaEconomizada", "VlrRcb",
"VlrDemandaReduzidaPonta")
```

```r
# Convertendo as colunas para numéricas, tratando os pontos e vírgulas
dados_completos[, colunas_numericas] <- lapply(dados_completos[, colunas_numericas], function(x)
as.numeric(gsub(",", ".", gsub("\\.", "", x))))

# Converter variáveis para os tipos corretos
dados_completos$DatGeracaoConjuntoDados <- as.Date(dados_completos$DatGeracaoConjuntoDados, format = "%Y-%m-%d")
dados_completos$DatInicioProjeto <- as.Date(dados_completos$DatInicioProjeto, format = "%Y-%m-%d")
dados_completos$DatConclusaoProjeto <- as.Date(dados_completos$DatConclusaoProjeto, format = "%Y-%m-%d")

# Visualizando novamente a estrutura dos dados
str(dados_completos)

# Tratamento de valores NA - Imputação com MICE

# Selecionando apenas algumas para imputação
# colunas_pmm <- c("VlrBeneficioEnergiaEconomizada", "VlrRcb", "VlrDemandaReduzidaPonta")
# dados_pmm <- dados_completos[, colunas_pmm]

# head(dados_pmm)

# # Imputação múltipla para lidar com valores ausentes
# imputed_data <- mice(dados_pmm, method = "pmm", seed = 500)
# dados_completos[, colunas_pmm] <- complete(imputed_data, 1)


# Criar coluna de duração do projeto (em dias)
dados_completos <- dados_completos %>%
  mutate(DuracaoProjeto = as.numeric(DatConclusaoProjeto - DatInicioProjeto))

# Visualizando novamente a estrutura dos dados
str(dados_completos)

# Resumo estatístico dos dados tratados
summary(dados_completos)

# 1. Contagem de valores nulos por coluna
colSums(is.na(dados_completos))

# 2. Porcentagem de valores nulos por coluna
colSums(is.na(dados_completos)) / nrow(dados_completos) * 100.0


### Analisando as colunas numéricas representativas de eficiência energética
### Percebemos os seguintes níveis de dados ausentes (NA):

### VlrBeneficioEnergiaEconomizada: 67.2%
### VlrRcb: 18.1%
### VlrDemandaReduzidaPonta: 0.5%

### Devido à extensa quantidade de dados ausentes na variável VlrBeneficioEnergiaEconomizada,
### optamos por não trabalhar com essa variável, de tal forma que vamos removê-la do conjunto dados_completos.

dados_completos <- dados_completos[, -c(which(colnames(dados_completos) == "VlrBeneficioEnergiaEconomizada"))]


### Analisando variáveis categóricas
### Objetivo: excluir variáveis categóricas com poucos dados da nossa análise


# Tabela de frequências para variáveis categóricas
table(dados_completos$DscTipologia)

# Resultado:

#    Aquecimento Solar            Baixa Renda             Co-geração
#                   35                   1599                      3
#  Comércio e Serviços Diagnóstico Energético           Educacional
#                 1937                      1                     13
#   Iluminação Pública             Industrial         Poder Público
#                 1007                    333                   2511
#          Prioritário         Projeto Piloto            Residencial
#                    4                     23                    658
#                Rural      Serviços Públicos
#                  102                    486

# Vamos agrupar as seguintes Tipologias por baixa disponibilidade de dados para análise como sendo "Outros"
# (menos de 100 exemplares):
```

```
#  - Aquecimento Solar
#  - Co-geração
#  - Diagnóstico Energético
#  - Educacional
#  - Prioritário
#  - Projeto piloto



### Analisando uso final

table(dados_completos$DscUsoFinal)

# Resultado:
#       Aquecimento             Aquecimento de Água
#                15                             387
#     Ar Comprimido          Condicionamento de Ar
#                15                             923
#      Força Motriz Geração por Fontes Incentivadas
#               339                             635
# Gestão Energética                     Iluminação
#                 1                            4870
#            Outros                     Reciclagem
#               734                              26
#      Refrigeração
#               767

# Vamos transformar todas as categorias de Uso Final com menos de 30 exemplares em "Outros":
#  - Aquecimento
#  - Ar comprimido
#  - Gestão energética
#  - Reciclagem




# Transformando tipologias com menos de 100 exemplares em "Outros"
tipologias_outros <- c("Aquecimento Solar", "Co-geração", "Diagnóstico Energético", "Educacional", "Prioritário",
"Projeto Piloto")
dados_completos$DscTipologia <- ifelse(dados_completos$DscTipologia %in% tipologias_outros, "Outros",
dados_completos$DscTipologia)

# Transformando categorias de Uso Final com menos de 30 exemplares em "Outros"
usos_finais_excluir <- c("Aquecimento", "Ar Comprimido", "Gestão Energética", "Reciclagem")
dados_completos$DscUsoFinal <- ifelse(dados_completos$DscUsoFinal %in% usos_finais_excluir, "Outros",
dados_completos$DscUsoFinal)


##################################
##### 4. Análise Descritiva #####
##################################

# Sumário estatístico
summary(dados_completos[, sapply(dados_completos, is.numeric)])

# Tabela de frequências para variáveis categóricas
table(dados_completos$DscTipologia)
table(dados_completos$DscUsoFinal)

# Medidas de dispersão e assimetria
Desc(dados_completos$VlrRcb)
Desc(dados_completos$VlrDemandaReduzidaPonta)
Desc(dados_completos$DuracaoProjeto)


#######################################
##### 5. Visualização dos Dados: #####
#######################################

# Histograma do retorno sobre o custo-benefício apenas valores diferentes de NA na coluna VlrRcb
# temp_data = dados_completos[!is.na(dados_completos$VlrRcb) & dados_completos$VlrRcb != 0, ]

# Função para remover outliers
# Essa função será utilizada para melhorar as visualizações dos dados
# Uma vez que outliers expandem a escala dos gráficos, dificultando a visualização
# Método utilizado: IQR (Interquartile Range)
remove_outliers <- function(data, column, multiplier = 1.5) {
  Q1 <- quantile(data[[column]], 0.25, na.rm = TRUE)
```

```r
  Q3 <- quantile(data[[column]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  # Definindo limites para detecção de outliers
  lower_bound <- Q1 - multiplier * IQR
  upper_bound <- Q3 + multiplier * IQR

  # Filtrando os dados para remover outliers
  data <- data %>%
    filter(data[[column]] >= lower_bound & data[[column]] <= upper_bound)

  return(data)
}

# Histograma do retorno sobre o custo-benefício (excluindo outliers)
temp_data <- remove_outliers(dados_completos, "VlrRcb")
ggplot(temp_data, aes(x = VlrRcb)) +
  geom_histogram(bins = 30, fill = "lightgreen", color = "black") +
  labs(title = "Histograma do Retorno sobre o Custo-Benefício", x = "Retorno (R$)", y = "Frequência")

# Histograma da demanda reduzida no pico (excluindo outliers)
temp_data <- remove_outliers(dados_completos, "VlrDemandaReduzidaPonta")
ggplot(temp_data, aes(x = VlrDemandaReduzidaPonta)) +
  geom_histogram(bins = 30, fill = "lightgreen", color = "black") +
  labs(title = "Histograma da Demanda Reduzida no Pico", x = "Demanda (kW)", y = "Frequência")

# Boxplot da duração do projeto por tipologia
temp_data <- remove_outliers(dados_completos, "DuracaoProjeto")
ggplot(temp_data, aes(x = DscTipologia, y = DuracaoProjeto)) +
  geom_boxplot() +
  labs(title = "Duração do Projeto por Tipologia", x = "Tipologia", y = "Duração (dias)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Boxplot do retorno sobre o custo-benefício por tipologia (excluindo outliers)
temp_data <- remove_outliers(dados_completos, "VlrRcb")
ggplot(temp_data, aes(x = DscTipologia, y = VlrRcb)) +
  geom_boxplot() +
  labs(title = "Retorno sobre o Custo-Benefício por Tipologia", x = "Tipologia", y = "Retorno (R$)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Boxplot da demanda reduzida no pico por tipologia (excluindo outliers)
temp_data <- remove_outliers(dados_completos, "VlrDemandaReduzidaPonta")
ggplot(temp_data, aes(x = DscTipologia, y = VlrDemandaReduzidaPonta)) +
  geom_boxplot() +
  labs(title = "Demanda Reduzida no Pico por Tipologia", x = "Tipologia", y = "Demanda (kW)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Gráfico de barras da tipologia do projeto
ggplot(dados_completos, aes(x = DscTipologia)) +
  geom_bar(fill = "lightgreen", color = "black") +
  labs(title = "Tipologias dos Projetos", x = "Tipologia", y = "Contagem") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Scatter plot (exemplo)
ggplot(dados_completos, aes(x = VlrBeneficioEnergiaEconomizada, y = DuracaoProjeto)) +
  geom_point() +
  labs(title = "Relação entre Benefício e Duração do Projeto", x = "Benefício (R$)", y = "Duração (dias)")

# Gráficos para atender aos objetivos:


# Gráfico 1.1.
# Boxplot do retorno sobre o custo-benefício por tipologia (excluindo outliers) - Atende ao objetivo 1 e 3
temp_data <- remove_outliers(dados_completos, "VlrRcb")
ggplot(temp_data, aes(x = DscTipologia, y = VlrRcb)) +
  geom_boxplot() +
  labs(title = "Retorno sobre o Custo-Benefício por Tipologia", x = "Tipologia", y = "Retorno (R$)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_hline(yintercept = mean(temp_data$VlrRcb, na.rm = TRUE), linetype = "dashed", color = "red")


# Gráfico 1.2.
# Boxplot da demanda reduzida no pico por tipologia (excluindo outliers) - Atende ao objetivo 1
temp_data <- remove_outliers(dados_completos, "VlrDemandaReduzidaPonta")
```

```r
ggplot(temp_data, aes(x = DscTipologia, y = VlrDemandaReduzidaPonta)) +
  geom_boxplot() +
  labs(title = "Demanda Reduzida no Pico por Tipologia", x = "Tipologia", y = "Demanda (kW)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_hline(yintercept = mean(temp_data$VlrDemandaReduzidaPonta, na.rm = TRUE), linetype = "dashed", color =
"red")


# Gráfico 2.1.
# Gráfico de barras da tipologia do projeto - contribui para objetivo 2
ggplot(dados_completos, aes(x = DscTipologia)) +
  geom_bar(fill = "lightgreen", color = "black") +
  labs(title = "Frequência das Tipologias de Projeto", x = "Tipologia", y = "Contagem") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) # Adiciona contagem em cada barra

# Gráfico 2.2.
# Gráfico de barras para DscUsoFinal - contribui para objetivo 2
ggplot(dados_completos, aes(x = DscUsoFinal)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Uso Final da Energia nos Projetos", x = "Uso Final", y = "Contagem") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) # Adiciona contagem em cada barra


# Gráfico 3.1
# Scatter plot para analisar a relação entre retorno e demanda reduzida (excluindo outliers) - Atende objetivo 3
temp_data <- dados_completos %>%
  filter(!is.na(VlrRcb) & !is.na(VlrDemandaReduzidaPonta)) %>%
  remove_outliers("VlrRcb") %>%
  remove_outliers("VlrDemandaReduzidaPonta")
ggplot(temp_data, aes(x = VlrRcb, y = VlrDemandaReduzidaPonta)) +
  geom_point() +
  labs(title = "Relação entre Retorno e Demanda Reduzida", x = "Retorno (R$)", y = "Demanda Reduzida (kW)") +
  geom_smooth(method = "lm", se = FALSE, color = "red") # Adiciona linha de regressão


####################################
##### 5. Análise de Correlação #####
####################################

# Matriz de correlação para variáveis numéricas
cor(dados_completos[, sapply(dados_completos, is.numeric)], use = "pairwise.complete.obs")

# Visualizacão da matriz de correlação em mapa de calor
ggcorr(data = dados_completos[, sapply(dados_completos, is.numeric)])

# Concluímos que os dados apresentados não mostram grande correlação entre as variáveis numéricas.


####################################
##### 6. Resultados da análise #####
####################################


# Sobre as Tipologias

# As tipologias de projetos que apresentam maior e menor retorno sobre o investimento (ROI)


# Auxiliados pelo gráfico 1.1. e pelo seguinte cálculo:
# Calculando o ROI médio por tipologia em ordem decrescente
temp_data <- remove_outliers(dados_completos, "VlrRcb")
roi_por_tipologia <- temp_data %>%
  group_by(DscTipologia) %>%
  summarise(ROI_Medio = mean(VlrRcb, na.rm = TRUE)) %>%
  arrange(desc(ROI_Medio))

# Mostrando os ROIs médios por tipologia
print(roi_por_tipologia)

# As tipologias de projetos que apresentam maior retorno sobre o investimento (ROI)
#  - Poder Público: média de 0.628
#  - Serviços Públicos: 0.623
#  - Outros (incluem as seguintes tipologias: Aquecimento Solar, Co-geração, Diagnóstico Energético, Educacional,
Prioritário, Projeto piloto): média de 0.619
#  - Comércio e Serviços: média de 0.614
```

```r
# As tipologias de projetos que apresentam menor retorno sobre o investimento (ROI)
#  - Iluminação Pública: média de 0.321
#  - Rural: média de 0.493


# As tipologias de projetos que apresentam maior e menor redução na demanda

# Auxiliados pelo gráfico 1.2. e pelo seguinte cálculo:
# Calculando a redução média na demanda por tipologia em ordem decrescente
temp_data <- remove_outliers(dados_completos, "VlrDemandaReduzidaPonta")
reducao_demanda_por_tipologia <- temp_data %>%
  group_by(DscTipologia) %>%
  summarise(RedDemanda_Media = mean(VlrDemandaReduzidaPonta, na.rm = TRUE)) %>%
  arrange(desc(RedDemanda_Media))

# Mostrando a redução média na demanda por tipologia
print(reducao_demanda_por_tipologia)

# As tipologias de projetos que apresentam maior redução na demanda
#  - Iluminação Pública: média de 67.8 kW
#  - Baixa Renda: média de 62.8 kW

# As tipologias de projetos que apresentam menor redução na demanda
#  - Comércio e Serviços: média de 25.9 kW
#  - Poder Público: média de 31.1 kW


# Sobre as Lacunas

# Auxiliados pelo Gráfico 2.1. e pelo seguinte cálculo
# Contagem de projetos por tipologia
contagem_tipologia <- dados_completos %>%
  group_by(DscTipologia) %>%
  summarise(Contagem = n()) %>%
  arrange(desc(Contagem))

print(contagem_tipologia)

# Tipologias com mais projetos
#  - Poder Público: 2511 projetos
#  - Comércio e Serviços: 1937 projetos
#  - Baixa renda: 1599 projetos

# Tipologias com menos projetos
#  - Outros (incluem as seguintes tipologias: Aquecimento Solar, Co-geração, Diagnóstico Energético, Educacional,
Prioritário, Projeto piloto): 79 projetos
#  - Rural: 102 projetos


# Auxiliados pelo Gráfico 2.2. e pelo seguinte cálculo
# Contagem de projetos por uso final
contagem_uso_final <- dados_completos %>%
  group_by(DscUsoFinal) %>%
  summarise(Contagem = n()) %>%
  arrange(desc(Contagem))

print(contagem_uso_final)

# Uso final com mais projetos
#  - Iluminação: 4870 projetos
#  - Condicionamento de Ar: 923 projetos

# Uso final com menos projetos
#  - Força Motriz: 339 projetos
#  - Aquecimiento de Água: 387 projetos



########################################
##### 7. Conclusão e Recomendações #####
########################################


# Conclusões:

# Análise de Retorno sobre Investimento (ROI) e Redução de Demanda:
# A análise revelou uma grande variação no ROI entre as diferentes tipologias de projetos de eficiência
energética.
```

```
# Projetos do setor público ("Poder Público" e "Serviços Públicos") apresentaram, em média, os maiores retornos,
# sugerindo que investimentos nessas áreas podem ser particularmente eficazes.
# Por outro lado, projetos de "Iluminação Pública" mostraram um ROI significativamente menor,
# indicando a necessidade de uma revisão estratégica nesse setor.

# Em relação à redução da demanda, "Iluminação Pública" e "Baixa Renda" se destacaram,
# enquanto "Comércio e Serviços" e "Poder Público" apresentaram reduções menores,
# sugerindo que as estratégias empregadas nestes últimos setores podem necessitar de aprimoramentos.
# A discrepância entre o alto ROI do setor público e sua relativamente baixa redução de demanda
# sugere que outros fatores além da redução de consumo direto contribuem para o retorno do investimento nesses
projetos.


# Identificação de Lacunas:  A análise de frequência das tipologias de projetos destaca um desequilíbrio na
alocação de recursos.
# Há uma concentração significativa de projetos em "Poder Público", "Comércio e Serviços" e "Baixa Renda",
# enquanto outras categorias ("Outros", "Rural") recebem uma atenção consideravelmente menor.
# Essa concentração pode indicar a necessidade de expandir o escopo do programa para alcançar outros setores e
contextos,
# promovendo uma maior equidade na distribuição dos benefícios da eficiência energética.
# Da mesma forma, a análise do uso final da energia revela uma preponderância de projetos focados em iluminação,
# enquanto outros usos finais, como força motriz e aquecimento de água, têm menor representatividade,
# indicando áreas potenciais para expansão e diversificação das ações de eficiência energética.


# Recomendações:

# 1. Investigação Adicional sobre Projetos de Iluminação Pública: O baixo retorno sobre o investimento em
projetos de iluminação pública
# justifica uma investigação detalhada sobre as causas dessa baixa performance.
# Isso inclui a análise de custos, métodos de implementação, tecnologias empregadas e critérios de seleção de
projetos.

# 2. Diversificação de Investimentos:
# É recomendado ampliar o alcance do programa de eficiência energética,
# direcionando mais recursos para tipologias e usos finais atualmente sub-representados,
# como projetos rurais e aqueles focados em setores além de iluminação e condicionamento de ar.
# Isso promoverá uma maior equidade na distribuição dos benefícios e otimizará o impacto do programa.

# 3. Análise de Dados Mais Granulares:
# Para uma análise mais robusta, a coleta de dados mais detalhados, incluindo informações geográficas,
# custos específicos dos projetos e indicadores de desempenho mais abrangentes,
# é crucial para refinar as estratégias e otimizar a eficácia dos investimentos.

# 4. Monitoramento e Avaliação Contínua:
# A implementação de um sistema de monitoramento e avaliação contínuo do programa de eficiência energética
# é essencial para acompanhar o progresso, identificar potenciais problemas e ajustar as estratégias conforme
necessário.
# Isso permitirá o aprimoramento contínuo do programa e o alcance de resultados mais efetivos.


# Considerações Finais:
# Esta análise exploratória forneceu valiosas informações sobre os padrões e tendências nos projetos de
eficiência energética.
# No entanto, é importante considerar as limitações dos dados disponíveis,
# especialmente a falta de informações geográficas e a alta porcentagem de valores ausentes em algumas variáveis.
# Estudos futuros com dados mais completos e detalhados poderão fornecer uma compreensão ainda mais aprofundada
do tema
# e subsidiar a formulação de políticas públicas mais eficazes.
```

Cognitive Data Science (CDS)

Sobre o Projeto

Este projeto faz parte da Global Solution, focando no desenvolvimento de uma solução baseada em Data Science, IoT e Python para otimizar o consumo de energia em diferentes ambientes (residenciais, comerciais e urbanos). O objetivo principal é melhorar a eficiência energética através da análise de dados históricos e integração com fontes renováveis.

Objetivos

- Analisar tendências históricas de consumo de energia elétrica
- Criar pipeline de dados para filtrar informações específicas do Brasil
- Avaliar demanda energética e consumo per capita
- Identificar padrões de consumo para otimização
- Integrar análise com sistemas IoT para tomada de decisão em tempo real

Dados Disponíveis

Estrutura dos Dados

O conjunto de dados inclui as seguintes dimensões principais:

1. **Dimensão Temporal**
    - Data (AAAAMMDD)
2. **Tipo de Consumidor**
    - Cativo
    - Livre
3. **Localização**
    - Sistema (Subsistema ou Sistemas Isolados)
    - UF (Unidade da Federação)
4. **Classificação de Consumo**
    - Setor Econômico (3 níveis)
    - Tensão de Fornecimento (3 níveis)
    - Faixas de Consumo (2 níveis)
5. **Métricas**
    - Número de Consumidores
    - Consumo em MWh

Detalhes Específicos

- **Tensão de Fornecimento**:
    - Alta Tensão: ≥ 69kV
    - Baixa Tensão: ≤ 1kV
- **Faixas de Consumo para Baixa Tensão**:
    - Convencional: 0-30 kWh, 31-100 kWh, 101-200 kWh, 201-300 kWh, 301-400 kWh, 401-500 kWh, 501-1000 kWh, > 1000 kWh
    - Baixa Renda: 0-30 kWh, 31-100 kWh, 101-200 kWh, > 200 kWh

Fontes de Dados

1. **Dados Processados**:
    - Consumo de Energia Elétrica - EPE
2. **Dados Brutos e Dicionário**:
    - Anuário Estatístico de Energia Elétrica - EPE

Tecnologias Utilizadas

- Python para processamento e análise de dados
- Banco de dados relacional para armazenamento
- Ferramentas de IoT para coleta de dados em tempo real
- Bibliotecas de Data Science para análise preditiva

Análises Previstas

1. Tendências de consumo ao longo do tempo
2. Distribuição por setor econômico
3. Análise comparativa entre regiões
4. Impacto das faixas de tensão no consumo
5. Relação entre número de consumidores e consumo total
6. Padrões sazonais de consumo

Entregáveis Esperados

- Pipeline de dados automatizado
- Análises estatísticas do consumo
- Dashboards interativos
- Modelos preditivos de consumo

- Documentação técnica completa

Configuração do Banco de Dados

Pré-requisitos

- PostgreSQL instalado
- Privilégios de administrador no banco de dados
- psql ou outro cliente SQL compatível com PostgreSQL

Scripts de Configuração:

O banco de dados é configurado em duas etapas, utilizando os seguintes scripts:

1. `src/initialize_database.sql`: Cria a estrutura inicial do banco de dados
2. `src/create_pipeline_views.sql`: Configura as views para análise dos dados

Passo 1: Criar Estrutura do Banco de Dados

Execute o script `src/initialize_database.sql`:

```
-- Criação das tabelas dimensionais
CREATE TABLE DIM_TEMPO (
    data DATE PRIMARY KEY,
    ano INT,
    mes INT,
    trimestre INT,
    mes_nome VARCHAR(20),
    CONSTRAINT chk_mes CHECK (mes BETWEEN 1 AND 12),
    CONSTRAINT chk_trimestre CHECK (trimestre BETWEEN 1 AND 4)
);

CREATE TABLE DIM_LOCALIZACAO (
    id_localizacao SERIAL PRIMARY KEY,
    uf VARCHAR(2),
    sistema VARCHAR(50),
    regiao VARCHAR(20),
    populacao DECIMAL(12,2),
    CONSTRAINT chk_uf CHECK (uf ~ '^[A-Z]{2}$')
);

CREATE TABLE DIM_CONSUMIDOR (
    id_consumidor SERIAL PRIMARY KEY,
    tipo_consumidor VARCHAR(10),
    faixa_consumo_n1 VARCHAR(50),
    faixa_consumo_n2 VARCHAR(50),
    CONSTRAINT chk_tipo_consumidor CHECK (tipo_consumidor IN ('Cativo', 'Livre'))
);

CREATE TABLE DIM_TENSAO (
    id_tensao SERIAL PRIMARY KEY,
    tensao_n1 VARCHAR(50),
    tensao_n2 VARCHAR(50),
    tensao_n3 VARCHAR(50)
);

CREATE TABLE DIM_SETOR (
    id_setor SERIAL PRIMARY KEY,
    setor_n1 VARCHAR(50),
    setor_n2 VARCHAR(50),
    setor_n3 VARCHAR(50)
);

-- Criação da tabela fato
CREATE TABLE FATO_CONSUMO (
    id_consumo SERIAL PRIMARY KEY,
    data DATE,
    id_localizacao INT,
    id_consumidor INT,
    id_tensao INT,
    id_setor INT,
    consumo_mwh DECIMAL(12,2),
    numero_consumidores INT,
    FOREIGN KEY (data) REFERENCES DIM_TEMPO(data),
    FOREIGN KEY (id_localizacao) REFERENCES DIM_LOCALIZACAO(id_localizacao),
```

```
    FOREIGN KEY (id_consumidor) REFERENCES DIM_CONSUMIDOR(id_consumidor),
    FOREIGN KEY (id_tensao) REFERENCES DIM_TENSAO(id_tensao),
    FOREIGN KEY (id_setor) REFERENCES DIM_SETOR(id_setor)
);

-- Índices para otimização de consultas
CREATE INDEX idx_fato_consumo_data ON FATO_CONSUMO(data);
CREATE INDEX idx_fato_consumo_localizacao ON FATO_CONSUMO(id_localizacao);
CREATE INDEX idx_fato_consumo_setor ON FATO_CONSUMO(id_setor);
```

Passo 2: Criar Views de Análise

Execute o script `src/create_pipeline_views.sql`:

```
-- Criação de views para análises específicas

-- View para análise de tendência de consumo mensal
CREATE VIEW vw_tendencia_consumo_mensal AS
SELECT
    dt.ano,
    dt.mes,
    dl.uf,
    dl.sistema,
    SUM(fc.consumo_mwh) as consumo_total,
    SUM(fc.numero_consumidores) as total_consumidores
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, dt.mes, dl.uf, dl.sistema
ORDER BY dt.ano, dt.mes;

-- View para cálculo do consumo per capita por UF
CREATE VIEW vw_consumo_per_capita AS
SELECT
    dt.ano,
    dl.uf,
    SUM(fc.consumo_mwh) as consumo_total,
    MAX(dl.populacao) as populacao,
    ROUND(SUM(fc.consumo_mwh) / MAX(dl.populacao), 2) as consumo_per_capita
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, dl.uf
ORDER BY dt.ano, dl.uf;

-- View para análise de demanda por setor econômico
CREATE VIEW vw_demanda_setor AS
SELECT
    dt.ano,
    ds.setor_n1,
    ds.setor_n2,
    dl.regiao,
    SUM(fc.consumo_mwh) as consumo_total,
    COUNT(DISTINCT fc.id_consumidor) as num_consumidores
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_SETOR ds ON fc.id_setor = ds.id_setor
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, ds.setor_n1, ds.setor_n2, dl.regiao
ORDER BY dt.ano, consumo_total DESC;
```

Explicação das Views de Análise

O script `create_pipeline_views.sql` cria três views principais para análise dos dados. Abaixo está a explicação detalhada de cada uma:

1. View `vw_tendencia_consumo_mensal`

```
CREATE VIEW vw_tendencia_consumo_mensal AS
SELECT
    dt.ano,
    dt.mes,
    dl.uf,
```

```
    dl.sistema,
    SUM(fc.consumo_mwh) as consumo_total,
    SUM(fc.numero_consumidores) as total_consumidores
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, dt.mes, dl.uf, dl.sistema
ORDER BY dt.ano, dt.mes;
```

Retorno:

Coluna	Tipo	Descrição
ano	INT	Ano de referência
mes	INT	Mês de referência (1-12)
uf	VARCHAR(2)	Sigla do estado
sistema	VARCHAR(50)	Nome do subsistema ou sistema isolado
consumo_total	DECIMAL	Soma total do consumo em MWh no período
total_consumidores	INT	Número total de consumidores no período

Uso: Análise de tendências mensais de consumo por estado e sistema elétrico.

2. View `vw_consumo_per_capita`

```
CREATE VIEW vw_consumo_per_capita AS
SELECT
    dt.ano,
    dl.uf,
    SUM(fc.consumo_mwh) as consumo_total,
    MAX(dl.populacao) as populacao,
    ROUND(SUM(fc.consumo_mwh) / MAX(dl.populacao), 2) as consumo_per_capita
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, dl.uf
ORDER BY dt.ano, dl.uf;
```

Retorno:

Coluna	Tipo	Descrição
ano	INT	Ano de referência
uf	VARCHAR(2)	Sigla do estado
consumo_total	DECIMAL	Consumo total em MWh
populacao	DECIMAL	População do estado
consumo_per_capita	DECIMAL	Consumo médio por habitante (MWh/habitante)

Uso: Análise comparativa do consumo de energia por habitante entre diferentes estados.

3. View `vw_demanda_setor`

```
CREATE VIEW vw_demanda_setor AS
SELECT
    dt.ano,
    ds.setor_n1,
    ds.setor_n2,
    dl.regiao,
    SUM(fc.consumo_mwh) as consumo_total,
    COUNT(DISTINCT fc.id_consumidor) as num_consumidores
FROM FATO_CONSUMO fc
JOIN DIM_TEMPO dt ON fc.data = dt.data
JOIN DIM_SETOR ds ON fc.id_setor = ds.id_setor
JOIN DIM_LOCALIZACAO dl ON fc.id_localizacao = dl.id_localizacao
GROUP BY dt.ano, ds.setor_n1, ds.setor_n2, dl.regiao
ORDER BY dt.ano, consumo_total DESC;
```

Retorno:

Coluna	Tipo	Descrição
ano	INT	Ano de referência
setor_n1	VARCHAR(50)	Classificação primária do setor econômico
setor_n2	VARCHAR(50)	Classificação secundária do setor econômico
regiao	VARCHAR(20)	Região geográfica
consumo_total	DECIMAL	Consumo total em MWh do setor
num_consumidores	INT	Número de consumidores únicos no setor

Uso: Análise setorial do consumo de energia, permitindo identificar os setores econômicos com maior demanda por região.

Exemplos de Consultas

```
-- Exemplo 1: Consumo mensal total em 2023 para São Paulo
SELECT * FROM vw_tendencia_consumo_mensal
WHERE ano = 2023 AND uf = 'SP';

-- Exemplo 2: Top 5 estados com maior consumo per capita em 2023
SELECT * FROM vw_consumo_per_capita
WHERE ano = 2023
ORDER BY consumo_per_capita DESC
LIMIT 5;

-- Exemplo 3: Consumo por setor industrial na região Sudeste
SELECT * FROM vw_demanda_setor
WHERE setor_n1 = 'Industrial'
AND regiao = 'Sudeste'
ORDER BY consumo_total DESC;
```

Como Executar os Scripts

1. Abra o terminal ou prompt de comando
2. Conecte-se ao PostgreSQL usando psql:

```
psql -U seu_usuario -d seu_banco
```

3. Execute os scripts na ordem correta:

```
\i caminho/para/initialize_database.sql
\i caminho/para/create_pipeline_views.sql
```

Verificação da Instalação

Para verificar se tudo foi instalado corretamente, execute:

```
-- Verificar tabelas criadas
\dt

-- Verificar views criadas
\dv

-- Testar uma view
SELECT * FROM vw_tendencia_consumo_mensal LIMIT 5;
```

Notas Importantes

- Certifique-se de ter backup dos dados antes de executar os scripts
- Os scripts devem ser executados com privilégios de administrador
- As constraints garantem a integridade dos dados
- Os índices otimizam as consultas mais frequentes
- As views facilitam a análise dos dados

- Os scripts podem ser adaptados conforme necessidade

Conclusão

Este projeto de análise de consumo de energia elétrica foi desenvolvido como parte da Global Solution, com foco em criar uma solução baseada em Data Science para otimização do consumo energético. A estrutura implementada oferece:

📊 Principais Funcionalidades

- Análise granular do consumo de energia por região, setor e período
- Acompanhamento de tendências de consumo ao longo do tempo
- Cálculo de métricas per capita para comparações entre regiões
- Base para integração com sistemas IoT e análise em tempo real

💡 Benefícios

1. **Tomada de Decisão:**
    - Insights baseados em dados históricos
    - Identificação de padrões de consumo
    - Suporte a decisões de eficiência energética
2. **Sustentabilidade:**
    - Monitoramento do impacto de iniciativas de economia
    - Base para integração com fontes renováveis
    - Suporte a políticas de redução de consumo
3. **Escalabilidade:**
    - Estrutura modular e expansível
    - Preparado para inclusão de novos dados
    - Facilidade de integração com outras ferramentas

🚀 Próximos Passos

- Implementação de modelos preditivos
- Integração com sistemas IoT em tempo real
- Desenvolvimento de dashboards interativos
- Expansão para análise de fontes renováveis
- Inclusão de variáveis climáticas e sazonais

📈 Impacto Esperado

O sistema desenvolvido tem potencial para contribuir significativamente com:

- Redução do consumo energético
- Otimização de recursos
- Promoção da sustentabilidade
- Conscientização sobre uso de energia
- Suporte a políticas públicas de eficiência energética

Este projeto estabelece uma base sólida para o desenvolvimento de soluções mais avançadas em eficiência energética, combinando análise de dados históricos com potencial para integração com tecnologias modernas de IoT e automação.

Computational Thinking with Python (CTWP)

Introdução

Este projeto foi desenvolvido como parte da atividade **Global Solutions (GS) 2024.2**, cujo tema é **Energia**. A solução utiliza **Data Science**, **IoT**, **Python** e **Banco de Dados** para otimizar o consumo de energia em ambientes residenciais, com foco na eficiência energética, sustentabilidade e redução de custos.

A aplicação apresentada oferece uma interface gráfica para monitoramento e gerenciamento do consumo energético, integrando dispositivos residenciais e simulando dados em tempo real. **Os dados de consumo simulados são gravados em um banco de dados PostgreSQL. Esses dados serão utilizados na integração completa do sistema na seção "Ir Além", conectando o sistema de monitoramento com outras partes do projeto (AICSS, CDS, SCR).** A ideia é proporcionar uma base funcional que pode ser expandida para incluir recursos avançados, como integração com fontes renováveis (solar, eólica) e análise preditiva.

Objetivos

- **Monitorar o consumo energético em tempo real:** Simular dispositivos e seu impacto no consumo total.
- **Promover eficiência energética:** Apresentar dados que ajudam o usuário a identificar padrões e reduzir custos.
- **Fornecer uma base escalável:** Preparar o sistema para integração com banco de dados e fontes renováveis.

- **Estimar custos de energia:** Calcular o custo com base no consumo diário e na tarifa energética.
- **Gravar dados em banco de dados:** Armazenar os dados de consumo para posterior análise e integração com outros módulos do projeto.

Funcionalidades do Sistema

1. Interface Gráfica

- Desenvolvida com **Tkinter**, apresentando um painel principal dividido em:
  - **Dispositivos:** Lista de dispositivos com status, potência e consumo.
  - **Gráficos:** Exibição do consumo energético em tempo real.
  - **Estatísticas:** Dados de consumo atual, diário e custo estimado.

2. Monitoramento em Tempo Real

- Atualizações automáticas a cada 10 segundos.
- Consumo de dispositivos simulados com variações aleatórias para refletir condições reais.

3. Gráficos Interativos

- Exibição do histórico de consumo com timestamps.
- Representação visual clara e intuitiva para análise do usuário.

4. Estatísticas de Consumo e Custo

- Consumo atual em watts (W).
- Consumo diário estimado em kilowatt-hora (kWh).
- Cálculo do custo com base na tarifa simulada (R$ 0,75/kWh).

5. (Opcional) Integração com Banco de Dados PostgreSQL

- Os dados de consumo de cada dispositivo, incluindo timestamp, potência (kW), frequência de atualização (em segundos), nome do dispositivo e status (ligado/desligado) são gravados no banco de dados.

Detalhamento Técnico

Tecnologias Utilizadas

- **Python 3.9+**
- **Tkinter:** Criação da interface gráfica.
- **Matplotlib:** Gráficos interativos de consumo.
- **Pandas:** Manipulação e análise de dados (possível expansão futura).
- **Random:** Simulação de dados para dispositivos.
- **psycopg2:** Conexão com o banco de dados PostgreSQL.

Estrutura do Código

1. **Classe Principal (`EnergyMonitorSystem`):**
   - Gerencia dispositivos, consumo e atualização dos dados.
   - Responsável pela interface gráfica e integração dos componentes.
   - Inclui métodos para conectar ao banco de dados, salvar e recuperar dados.
2. **Simulação de Consumo:**
   - Estima o consumo de dispositivos com base no status atual (ligado/desligado) e uma variação aleatória.
3. **Atualização Automática:**
   - Usa `root.after()` para atualizar os dados e gráficos em intervalos regulares.
4. **(Opcional) Conexão e Gravação de Dados:**
   - Usa a biblioteca `psycopg2` para conectar a um banco de dados PostgreSQL e gravar os dados de consumo. A função `save_consumption_data()` realiza a gravação, utilizando transações para garantir a integridade dos dados.

Utilidade Prática

- **Residências:** Monitoramento detalhado do consumo energético e identificação de dispositivos com alto consumo.
- **Educação:** Demonstração de conceitos de IoT e otimização energética para estudantes e profissionais.
- **Base para Projetos Avançados:** Escalável para integração com fontes renováveis, banco de dados e dispositivos reais.

Possibilidades de Expansão

1. Integração com Banco de Dados (Já implementada)

- Armazenar histórico de consumo para análises futuras.
- Salvar configurações personalizadas de dispositivos.

2. Fontes de Energia Renovável

- Monitorar geração solar e eólica.
- Selecionar automaticamente a fonte mais econômica e sustentável.

3. Análise Preditiva

- Usar algoritmos de machine learning para prever padrões de consumo.
- Detectar anomalias e sugerir ações corretivas.

4. Relatórios e Notificações

- Gerar relatórios em PDF com estatísticas detalhadas.
- Enviar notificações em tempo real (e-mail ou push).

5. Controle Automático

- Ativar/desativar dispositivos automaticamente com base em regras predefinidas (ex.: horários de pico).

Como Usar

1. **Pré-requisitos:**
   - Python 3.9+ instalado.
   - Bibliotecas necessárias: `tkinter`, `matplotlib`, `pandas`, `psycopg2-binary`. Instale com: `pip install tkinter matplotlib pandas psycopg2-binary`
   - PostgreSQL instalado e configurado. Um usuário e banco de dados devem ser criados (`fiap_gs` e `gs_energia_residencial`, respectivamente), com as permissões descritas na seção "Configuração do Banco de Dados".
   - Execute os comandos SQL para criação de tabela e concessão de permissões descritos na seção "Configuração do Banco de Dados".
2. **(Opcional) Configuração do Banco de Dados:**
   - O script SQL necessário para a criaçao de usuário, banco de dados e tabela está presente no diretório `src/CTWP/script/initialize_db.sql`.
   - Crie um usuário e um banco de dados no PostgreSQL:

     ```
     CREATE USER fiap_gs WITH PASSWORD 'fiap_gs';
     CREATE DATABASE gs_energia_residencial OWNER fiap_gs;
     \c gs_energia_residencial fiap_gs
     GRANT ALL PRIVILEGES ON DATABASE gs_energia_residencial TO fiap_gs;
     GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO fiap_gs;
     GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO fiap_gs;
     ```

   - Execute o código SQL acima para criar usuário, a database `gs_energia_residencial`, a tabela `CONSUMO_RESIDENCIAL` e todos os privilégios necessários.
3. **Execução:**
   - Clone este repositório:

     ```
     git clone https://github.com/brunoconterato/fiap_global_solutions_2024
     cd fiap_global_solutions_2024/src/CTWP
     ```

   - Execute o script principal:

     ```
     python main.py
     ```

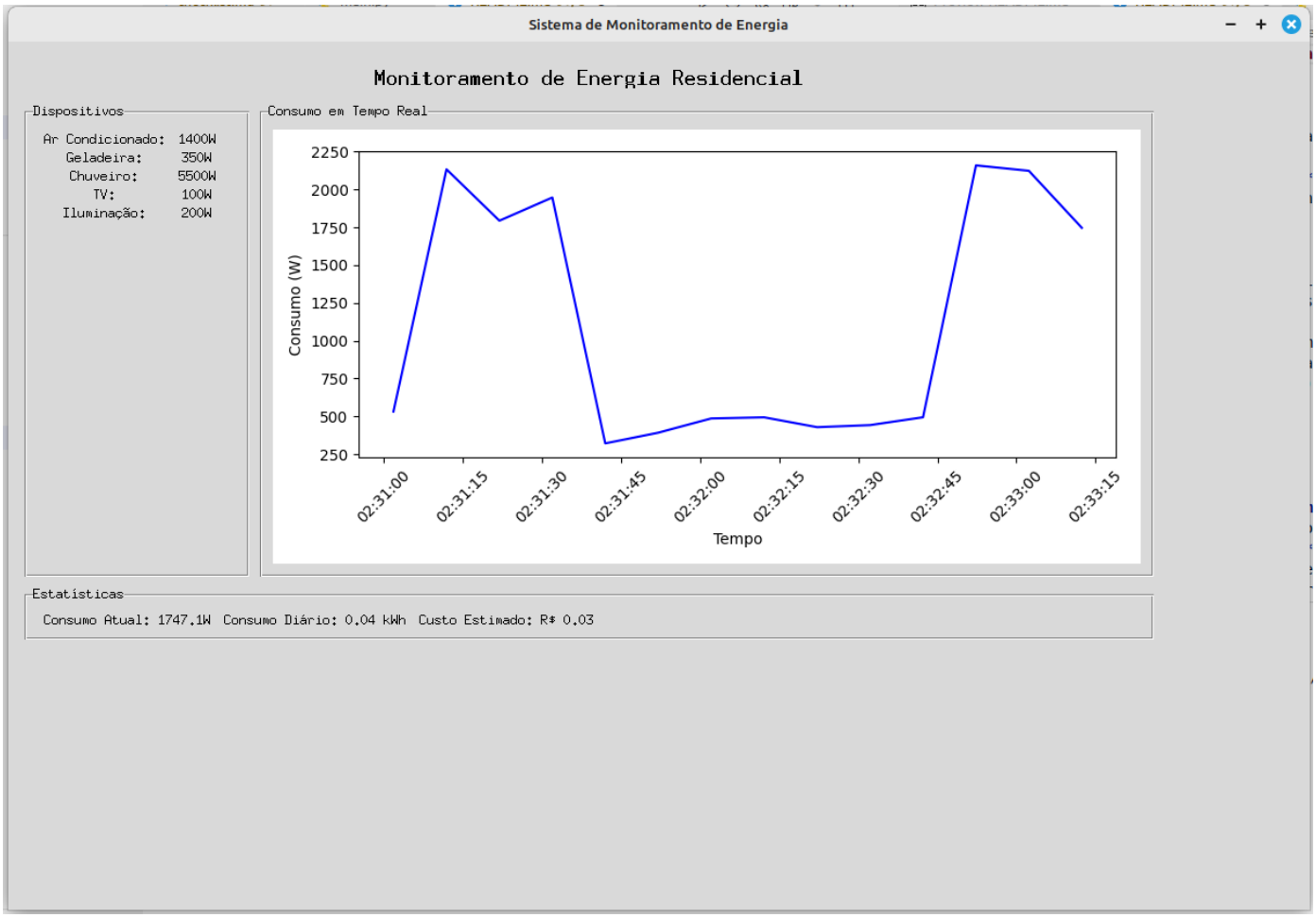
   - (Opcional) Substitua os valores das credenciais de conexão com o banco de dados PostgreSQL.
4. **Interface:**
   - Visualize os dispositivos e seu status.
   - Acompanhe o consumo em tempo real no gráfico.
   - Consulte as estatísticas de consumo e custo estimado.

Conclusão

Este projeto é um ponto de partida para o desenvolvimento de sistemas inteligentes de monitoramento energético. Ele combina conceitos de programação, ciência de dados e IoT, com foco na sustentabilidade e eficiência energética. A solução pode ser personalizada e expandida conforme as necessidades específicas de diferentes cenários residenciais, comerciais ou urbanos.

Apêndices

A. Imagem da Interface Gráfica



B. Código-fonte (O código está em `src/CTWP/src/main.py`.)

```
import tkinter as tk
from tkinter import ttk
import random
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
import psycopg2

class EnergyMonitorSystem:
    def __init__(self, root, db_host, db_port, db_name, db_user, db_password):
        self.initialize_db(db_host, db_port, db_name, db_user, db_password)

        self.root = root
        self.root.title("Sistema de Monitoramento de Energia")
        self.root.geometry("1200x800")

        # Dados simulados
        self.devices = {
            "Ar Condicionado": {"power": 1400, "status": "ON"},
            "Geladeira": {"power": 350, "status": "ON"},
            "Chuveiro": {"power": 5500, "status": "ON"},
            "TV": {"power": 100, "status": "ON"},
            "Iluminação": {"power": 200, "status": "ON"}
        }

        self.current_consumption = 0
        self.consumption_history = []
```

```
        self.timestamps = []

        self.setup_gui()
        self.update_data()

    def setup_gui(self):
        # Frame principal
        main_frame = ttk.Frame(self.root, padding="10")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        # Título
        title = ttk.Label(main_frame, text="Monitoramento de Energia Residencial",
                          font=('Helvetica', 16, 'bold'))
        title.grid(row=0, column=0, columnspan=2, pady=10)

        # Frame esquerdo - Dispositivos
        devices_frame = ttk.LabelFrame(main_frame, text="Dispositivos", padding="10")
        devices_frame.grid(row=1, column=0, padx=5, pady=5, sticky=(tk.W, tk.E, tk.N, tk.S))

        # Lista de dispositivos
        for i, (device, info) in enumerate(self.devices.items()):
            ttk.Label(devices_frame, text=f"{device}:").grid(row=i, column=0, padx=5, pady=2)
            ttk.Label(devices_frame, text=f"{info['power']}W").grid(row=i, column=1, padx=5, pady=2)
            status_var = tk.StringVar(value=info['status'])
            ttk.Label(devices_frame, textvariable=status_var).grid(row=i, column=2, padx=5, pady=2)

        # Frame direito - Gráficos
        graph_frame = ttk.LabelFrame(main_frame, text="Consumo em Tempo Real", padding="10")
        graph_frame.grid(row=1, column=1, padx=5, pady=5, sticky=(tk.W, tk.E, tk.N, tk.S))

        # Criar figura do matplotlib
        self.fig, self.ax = plt.subplots(figsize=(8, 4))
        self.canvas = FigureCanvasTkAgg(self.fig, master=graph_frame)
        self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

        # Frame inferior - Estatísticas
        stats_frame = ttk.LabelFrame(main_frame, text="Estatísticas", padding="10")
        stats_frame.grid(row=2, column=0, columnspan=2, padx=5, pady=5, sticky=(tk.W, tk.E))

        # Labels para estatísticas
        self.current_power_var = tk.StringVar(value="Consumo Atual: 0W")
        self.daily_consumption_var = tk.StringVar(value="Consumo Diário: 0 kWh")
        self.cost_var = tk.StringVar(value="Custo Estimado: R$ 0,00")

        ttk.Label(stats_frame, textvariable=self.current_power_var).grid(row=0, column=0, padx=5)
        ttk.Label(stats_frame, textvariable=self.daily_consumption_var).grid(row=0, column=1, padx=5)
        ttk.Label(stats_frame, textvariable=self.cost_var).grid(row=0, column=2, padx=5)

    def simulate_consumption(self):
        """Simula o consumo de energia dos dispositivos"""
        total = 0
        for device, info in self.devices.items():
            if info['status'] == 'ON':
                # Adiciona alguma variação aleatória
                variation = random.uniform(0.8, 1.2)
                total += info['power'] * variation
        return total

    def update_data(self):
        """Atualiza os dados e gráficos"""
        # Simular novo consumo
        self.current_consumption = self.simulate_consumption()
        self.consumption_history.append(self.current_consumption)
        self.timestamps.append(datetime.now())

        # Manter apenas os últimos 60 pontos (10 minutos)
        if len(self.consumption_history) > 60:
            self.consumption_history.pop(0)
            self.timestamps.pop(0)

        self.save_consumption_data() #Salva os dados no banco de dados.

        # Atualizar gráfico
        self.ax.clear()
        self.ax.plot(self.timestamps, self.consumption_history, 'b-')
        self.ax.set_xlabel('Tempo')
        self.ax.set_ylabel('Consumo (W)')
        self.ax.tick_params(axis='x', rotation=45)
```

```
        self.fig.tight_layout()
        self.canvas.draw()

        # Atualizar estatísticas
        self.current_power_var.set(f"Consumo Atual: {self.current_consumption:.1f}W")
        daily_kwh = sum(self.consumption_history) * 10 / (3600 * 1000)  # convertendo para kWh
        self.daily_consumption_var.set(f"Consumo Diário: {daily_kwh:.2f} kWh")
        cost = daily_kwh * 0.75  # Considerando tarifa de R$ 0,75 por kWh
        self.cost_var.set(f"Custo Estimado: R$ {cost:.2f}")

        # Simular mudança aleatória no status dos dispositivos
        for device in self.devices:
            if random.random() < 0.1:  # 10% de chance de mudar o status
                self.devices[device]['status'] = 'ON' if random.random() < 0.5 else 'OFF'

        # Agendar próxima atualização
        self.root.after(10000, self.update_data)  # Atualiza a cada 10 segundos

    def initialize_db(self, db_host, db_port, db_name, db_user, db_password):
        self.db_host = db_host
        self.db_port = db_port
        self.db_name = db_name
        self.db_user = db_user
        self.db_password = db_password
        self.conn = None
        self.cur = None
        self.connect_to_db()

    def connect_to_db(self):
        try:
            self.conn = psycopg2.connect(host=self.db_host, database=self.db_name, user=self.db_user,
password=self.db_password, port=self.db_port)
            self.cur = self.conn.cursor()
            print("Conectado ao banco de dados PostgreSQL")
        except psycopg2.Error as e:
            print(f"Erro ao conectar ao banco de dados: {e}")

    def close_db_connection(self):
        if self.conn:
            self.cur.close()
            self.conn.close()
            print("Conexão com o banco de dados fechada")

    def save_consumption_data(self):
        try:
            self.cur.execute("BEGIN;")  # inicia uma transação
            for device, info in self.devices.items():
                if info['status'] == 'ON':
                    consumo_kw = info['power'] / 1000.0  # convertendo para kW
                    try:
                        self.cur.execute("""
                            INSERT INTO CONSUMO_RESIDENCIAL (timestamp, consumo_potencia_kw,
frequencia_atualizacao_s, dispositivo, status)
                            VALUES (%s, %s, %s, %s, %s)
                        """, (datetime.now(), consumo_kw, 10, device, info['status']))
                    except psycopg2.IntegrityError as e:  # captura erro de chave única
                        if 'unique constraint' in str(e).lower():
                            print(f"Aviso: Tentativa de inserção duplicada ignorada para {device}.")
                        else:
                            raise e  # relança outras exceções

            self.conn.commit()  # commit da transação inteira, caso tudo tenha dado certo
            print("Dados de consumo salvos no banco de dados.")

        except psycopg2.Error as e:
            print(f"Erro ao salvar dados no banco de dados: {e}")
            self.conn.rollback()  # Rollback caso ocorra algum erro, mantendo a consistência
        except Exception as e:
            self.conn.rollback()
            print(f"Erro genérico: {e}")

if __name__ == "__main__":
    root = tk.Tk()

    #Credenciais do seu banco de dados
    db_host = "localhost" #ou seu IP
    db_port = "5432"
    db_name = "gs_energia_residencial"
```

```
    db_user = "fiap_gs"
    db_password = "fiap_gs"

    app = EnergyMonitorSystem(root, db_host, db_port, db_name, db_user, db_password)
    root.mainloop()
```

Ir Além

Este documento descreve a integração completa do sistema de gerenciamento de energia, conectando as diferentes partes do projeto: AICSS (sistema de iluminação com ESP32), CTWP (sistema de gerenciamento em Python), CDS (banco de dados), e SCR (análise estatística com R). O foco principal desta seção é a integração dos dados entre as diferentes partes do projeto, demonstrando a funcionalidade do sistema integrado.

Link para o Vídeo de Apresentação

Funcionamento

O sistema funciona da seguinte maneira:

1. **AICSS (ESP32):** O microcontrolador ESP32, utilizando sensores (ou botões simulados), monitora a luminosidade e presença (ou entradas de botões) e controla a iluminação interna e externa da residência. Os dados são simulados no Wokwi, e exportados manualmente (luminosidade, presença e status do LED) para um arquivo CSV.
2. **CTWP (Python):** O programa em Python lê os dados do arquivo CSV gerado pelo AICSS, simulando os dados de consumo de energia para os dispositivos controlados pelo sistema de iluminação. Ele também interage com o banco de dados PostgreSQL (CDS), gravando o consumo de energia dos dispositivos em tempo real e lendo dados históricos para gerar relatórios. Finalmente, a interface gráfica do sistema Python apresenta os dados de consumo, permitindo a visualização e análise.
3. **CDS (PostgreSQL):** O banco de dados PostgreSQL armazena os dados de consumo de energia coletados pelo sistema Python. Uma tabela específica (`CONSUMO_RESIDENCIAL`) foi criada para armazenar os dados de consumo residencial simulados.
4. **SCR (R):** (Opcional, dependendo do tempo disponível) A análise estatística usando R pode ser integrada posteriormente, lendo dados do banco de dados PostgreSQL para uma análise mais completa do consumo de energia e identificação de padrões.

Configuração Local do PostgreSQL

Para executar a aplicação localmente, você precisará ter o PostgreSQL instalado e configurado. As seguintes instruções assumem que você já possui o PostgreSQL instalado.

1. Crie o Usuário e Banco de Dados:

Esse script está localizado no diretório `src/CTWP/script/initialize_db.sql`.

Abra o console psql (com privilégios de administrador) e execute os seguintes comandos:

```
CREATE USER fiap_gs WITH PASSWORD 'fiap_gs';
CREATE DATABASE gs_energia_residencial OWNER fiap_gs;
```

2. Crie a Tabela CONSUMO_RESIDENCIAL:

Conecte-se ao banco de dados `gs_energia_residencial` como o usuário `fiap_gs` e execute o comando SQL abaixo:

```
\c gs_energia_residencial fiap_gs

-- Grant all privileges on the database to the user
GRANT ALL PRIVILEGES ON DATABASE gs_energia_residencial TO fiap_gs;


-- Create the table to store the simulated residential energy consumption data
CREATE TABLE CONSUMO_RESIDENCIAL (
    id_consumo SERIAL PRIMARY KEY,
    timestamp TIMESTAMP WITH TIME ZONE,
    consumo_potencia_kw DECIMAL(10,4),
    frequencia_atualizacao_s INT,
    dispositivo VARCHAR(50),
    status VARCHAR(10)
);
-- CREATE INDEX idx_consumo_residencial_timestamp ON CONSUMO_RESIDENCIAL(timestamp);

-- Grant all privileges on the table to the user
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO fiap_gs;
```

```
-- Grant usage on the sequence (this is the crucial part)
GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO fiap_gs;
```

3. Configure o Arquivo de Conexão do Python:

Modifique o arquivo `main.py` (ou o arquivo correspondente no seu projeto) para incluir as credenciais corretas para conectar ao seu banco de dados PostgreSQL:

```
# ... (código existente) ...
self.conn = psycopg2.connect(host="localhost", database="gs_energia_residencial", user="fiap_gs",
password="fiap_gs")
# ... (código existente) ...
```

Substitua `"localhost"` pelo nome do host do seu banco de dados se ele não estiver na mesma máquina.

4. Execução da Aplicação:

Após a configuração, execute o programa Python. O sistema irá gravar os dados simulados no banco de dados.

Dados de Entrada (AICSS Simulado)

O arquivo CSV com os dados simulados do AICSS (luminosidade, presença e status do LED) deve ser colocado em um local acessível ao script Python. O nome do arquivo e o caminho devem ser ajustados no script `main.py`. Um exemplo de arquivo CSV:

```
timestamp,consumo_potencia_kw,frequencia_atualizacao_s,dispositivo,status
10371,0.0010,5,led_externo_1,ligado_min
15397,0.0100,5,led_externo_1,ligado_max
20423,0.0100,5,led_externo_1,ligado_max
25449,0.0100,5,led_externo_1,ligado_max
30475,0.0100,5,led_externo_1,ligado_max
35501,0.0100,5,led_externo_1,ligado_max
35515,0.0100,5,led_interno_1,ligado
40528,0.0100,5,led_externo_1,ligado_max
45554,0.0100,5,led_externo_1,ligado_max
```

Cálculo de Energia Consumida

Cada registro na tabela CONSUMO_RESIDENCIAL representa o consumo de energia de um dispositivo em um determinado instante. Para calcular a energia total consumida por um dispositivo em um período, considere os seguintes fatores:

- consumo_potencia_kw: Potência consumida pelo dispositivo em kW no instante da medição.
- frequencia_atualizacao_s: Intervalo de tempo em segundos entre as medições.

A energia consumida (em kWh) para cada registro é calculada usando a fórmula:

Energia (kWh) = consumo_potencia_kw * frequencia_atualizacao_s / 3600

A energia total consumida por um dispositivo em um período é a soma das energias calculadas para cada registro desse dispositivo naquele período. A unidade resultante é kWh. Isso permite um cálculo mais preciso do consumo energético, considerando a frequência de atualização dos dados.

Código-fonte (O código está em `src/IrAlem/src/save_iot_data.py`. Dados simulados em `src/IrAlem/iot_data/data.csv`)

Código Python:

```
import pandas as pd
import psycopg2
import argparse
import os

def save_iot_data_to_db(csv_filepath, db_host, db_port, db_name, db_user, db_password):
    """
    Loads IoT data from a CSV file and saves it to a PostgreSQL database.

    Args:
        csv_filepath: Path to the CSV file.
        db_host: PostgreSQL database host.
        db_port: PostgreSQL database port.
        db_name: PostgreSQL database name.
        db_user: PostgreSQL database user.
        db_password: PostgreSQL database password.
    """
```

```
    conn = None
    try:
        # Load data from CSV
        csv_filepath = os.path.abspath(csv_filepath)
        df = pd.read_csv(csv_filepath)

        # Convert timestamp column from milliseconds to datetime objects
        df['timestamp'] = pd.to_datetime(df['timestamp'], unit='s')

        #Establish database connection
        conn = psycopg2.connect(host=db_host, database=db_name, user=db_user, password=db_password, port=db_port)
        cur = conn.cursor()

        # Insert data into database.  Using parameterized queries for security.
        for index, row in df.iterrows():
            cur.execute("""
                INSERT INTO CONSUMO_RESIDENCIAL (timestamp, consumo_potencia_kw, frequencia_atualizacao_s,
dispositivo, status)
                VALUES (%s, %s, %s, %s, %s)
            """, (row['timestamp'], row['consumo_potencia_kw'], row['frequencia_atualizacao_s'],
row['dispositivo'], row['status']))

        conn.commit()
        print(f"Dados do arquivo '{csv_filepath}' salvos com sucesso no banco de dados.")

    except FileNotFoundError:
        print(f"Erro: Arquivo CSV '{csv_filepath}' não encontrado.")
    except psycopg2.Error as e:
        print(f"Erro ao conectar ou gravar dados no banco de dados: {e}")
        conn.rollback()  # Rollback da transação em caso de erro
    except pd.errors.EmptyDataError:
        print(f"Erro: Arquivo CSV '{csv_filepath}' está vazio.")
    except pd.errors.ParserError:
        print(f"Erro: Erro ao analisar o arquivo CSV '{csv_filepath}'. Verifique o formato.")
    except Exception as e:
        print(f"Erro genérico: {e}")
    finally:
        if conn:
            cur.close()
            conn.close()


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Save IoT data from CSV to PostgreSQL.")
    parser.add_argument("--csv_filepath", default="src/IrAlem/iot_data/data.csv", help="Path to the CSV file.")
    parser.add_argument("--host", default="localhost", help="PostgreSQL database host.")
    parser.add_argument("--port", type=int, default=5432, help="PostgreSQL database port.")
    parser.add_argument("--dbname", default="gs_energia_residencial", help="PostgreSQL database name.")
    parser.add_argument("--user", default="fiap_gs", help="PostgreSQL database user.")
    parser.add_argument("--password", default="fiap_gs", help="PostgreSQL database password.")
    args = parser.parse_args()
    save_iot_data_to_db(args.csv_filepath, args.host, args.port, args.dbname, args.user, args.password)
```

Análise de Consumo de Energia Residencial

Este notebook realiza uma análise exploratória de dados de consumo de energia residencial. Utilizando dados extraídos de um banco de dados PostgreSQL, são realizadas diversas etapas de processamento e visualização para entender melhor os padrões de consumo de energia. As principais etapas incluem:

O objetivo é fornecer insights que possam ajudar na otimização do consumo de energia residencial.

1. Carregamento de Bibliotecas

Nesta seção, carregamos todas as bibliotecas necessárias para a análise de dados de consumo de energia residencial. As bibliotecas utilizadas incluem:

- `psycopg2` para conexão e extração de dados do banco de dados PostgreSQL.
- `pandas` para manipulação e análise de dados.
- `numpy` para operações numéricas.
- `matplotlib` e `seaborn` para visualização de dados.
- `datetime` para manipulação de datas e horários.

As bibliotecas são carregadas no início do notebook para garantir que todas as funcionalidades necessárias estejam disponíveis ao longo da análise.

```
import psycopg2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

2. Carregamento de Dados

Nesta seção, realizamos a conexão com o banco de dados PostgreSQL e extraímos os dados da tabela `CONSUMO_RESIDENCIAL`. Os dados são carregados em um DataFrame do pandas para posterior análise. As etapas incluem:

1. Estabelecer a conexão com o banco de dados utilizando as credenciais apropriadas.
2. Executar a consulta SQL para extrair todos os registros da tabela `CONSUMO_RESIDENCIAL`.
3. Carregar os dados extraídos em um DataFrame do pandas.
4. Realizar a conversão de tipos de dados e criar colunas adicionais para facilitar a análise.

O código correspondente para esta etapa está no próximo bloco de código.

```
# Conexão com o banco de dados PostgreSQL (substitua pelas suas credenciais)
conn = psycopg2.connect(
    dbname="gs_energia_residencial",
    user="fiap_gs",
    password="fiap_gs",
    host="localhost",
    port="5432"
)

# Consulta SQL para extrair dados da tabela
query = "SELECT * FROM CONSUMO_RESIDENCIAL"

# Executar a consulta SQL
cur = conn.cursor()
cur.execute(query)

# Obter os dados da consulta
rows = cur.fetchall()

# Definir os nomes das colunas
columns = ['id_consumo', 'timestamp', 'consumo_potencia_kw', 'frequencia_atualizacao_s', 'dispositivo', 'status']

# Carregar os dados em um DataFrame do pandas
df = pd.DataFrame(rows, columns=columns)
df = df.set_index('id_consumo')

df['timestamp'] = pd.to_datetime(df['timestamp'])
df['consumo_potencia_kw'] = df['consumo_potencia_kw'].astype(float)
df['energia_kwh'] = (df['consumo_potencia_kw'] * df['frequencia_atualizacao_s'] / 3600).round(4)

df['tempo_uso_h'] = (df['frequencia_atualizacao_s'] / 3600).round(4)

# Fechar o cursor
cur.close()
df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
```

```
    text-align: right;
}
```

	timestamp	consumo_potencia_kw	frequencia_atualizacao_s	dispositivo	status	energia_kwh	tempo_uso_h
id_consumo							
186	2024-11-25 20:40:49.756216-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
187	2024-11-25 20:40:49.757580-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
188	2024-11-25 20:40:49.757946-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
189	2024-11-25 20:40:49.759523-03:00	0.100	10	TV	ON	0.0003	0.0028
190	2024-11-25 20:40:49.759729-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
191	2024-11-25 20:40:59.897784-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
192	2024-11-25 20:40:59.898101-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
193	2024-11-25 20:40:59.898247-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
194	2024-11-25 20:40:59.898370-03:00	0.100	10	TV	ON	0.0003	0.0028
195	2024-11-25 20:40:59.898494-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
196	2024-11-25 20:41:09.977491-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
197	2024-11-25 20:41:09.977911-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
198	2024-11-25 20:41:09.978060-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
199	2024-11-25 20:41:09.978167-03:00	0.100	10	TV	ON	0.0003	0.0028
200	2024-11-25 20:41:09.978263-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
201	2024-11-25 20:41:20.095589-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
202	2024-11-25 20:41:20.096144-03:00	0.350	10	Geladeira	ON	0.0010	0.0028

	timestamp	consumo_potencia_kw	frequencia_atualizacao_s	dispositivo	status	energia_kwh	tempo_uso_h
id_consumo							
203	2024-11-25 20:41:20.096367-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
204	2024-11-25 20:41:20.096537-03:00	0.100	10	TV	ON	0.0003	0.0028
205	2024-11-25 20:41:20.096713-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
206	2024-11-25 20:41:30.193252-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
207	2024-11-25 20:41:30.193586-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
208	2024-11-25 20:41:30.193846-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
209	2024-11-25 20:41:30.194025-03:00	0.100	10	TV	ON	0.0003	0.0028
210	2024-11-25 20:41:30.194213-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
211	2024-11-25 20:41:40.310684-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
212	2024-11-25 20:41:40.311025-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
213	2024-11-25 20:41:40.311177-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
214	2024-11-25 20:41:40.311305-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
215	2024-11-25 20:41:50.396276-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
216	2024-11-25 20:41:50.397400-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
217	2024-11-25 20:41:50.397713-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
218	2024-11-25 20:41:50.397921-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
219	2024-11-25 20:42:00.492029-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
220	2024-11-25 20:42:00.492382-03:00	0.350	10	Geladeira	ON	0.0010	0.0028

	timestamp	consumo_potencia_kw	frequencia_atualizacao_s	dispositivo	status	energia_kwh	tempo_uso_h
id_consumo							
221	2024-11-25 20:42:00.492591-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
222	2024-11-25 20:42:00.492714-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
223	2024-11-25 20:42:10.591500-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
224	2024-11-25 20:42:10.591755-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
225	2024-11-25 20:42:10.591852-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
226	2024-11-25 20:42:10.592014-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
227	2024-11-25 20:42:20.696466-03:00	1.400	10	Ar Condicionado	ON	0.0039	0.0028
228	2024-11-25 20:42:20.696780-03:00	0.350	10	Geladeira	ON	0.0010	0.0028
229	2024-11-25 20:42:20.696962-03:00	5.500	10	Chuveiro	ON	0.0153	0.0028
230	2024-11-25 20:42:20.697075-03:00	0.200	10	Iluminação	ON	0.0006	0.0028
231	1970-01-01 11:19:00-03:00	0.001	5	led_externo_1	ligado_min	0.0000	0.0014
232	1970-01-01 12:42:45-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
233	1970-01-01 14:06:51-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
234	1970-01-01 14:07:05-03:00	0.010	5	led_interno_1	ligado	0.0000	0.0014
235	1970-01-01 15:30:37-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
236	1970-01-01 16:54:22-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
237	1970-01-01 18:18:07-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
238	1970-01-01 19:41:52-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
239	1970-01-01 21:05:37-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
240	1970-01-01 22:29:22-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
241	1970-01-01 22:29:36-03:00	0.010	5	led_interno_1	ligado	0.0000	0.0014

	timestamp	consumo_potencia_kw	frequencia_atualizacao_s	dispositivo	status	energia_kwh	tempo_uso_h
id_consumo							
242	1970-01-01 23:53:08-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
243	1970-01-01 23:53:22-03:00	0.010	5	led_interno_1	ligado	0.0000	0.0014
244	1970-01-02 01:16:54-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014
245	1970-01-02 02:40:39-03:00	0.010	5	led_externo_1	ligado_max	0.0000	0.0014

3. Análise Exploratória de Dados (EDA)

Nesta seção, realizamos a Análise Exploratória de Dados (EDA) para entender melhor os padrões de consumo de energia. As etapas incluem:

1. Sumário estatístico dos dados.
2. Verificação de valores ausentes.
3. Visualização da distribuição do consumo de energia.
4. Análise do consumo de energia por dispositivo.
5. Análise do tempo de uso por dispositivo.
6. Identificação e remoção de outliers.
7. Distribuição dos dispositivos.
8. Análise da relação entre consumo e tempo.

A EDA nos ajuda a identificar tendências, padrões e possíveis problemas nos dados, como valores ausentes ou outliers, que podem afetar a análise.

3.1. Sumário Estatístico

Nesta subseção, apresentamos um sumário estatístico dos dados de consumo de energia. O sumário inclui medidas como média, mediana, desvio padrão, valores mínimos e máximos, entre outras estatísticas descritivas. Essas informações são úteis para entender a distribuição e a variabilidade dos dados, bem como para identificar possíveis anomalias ou outliers.

```
# Análise Exploratória de Dados

# 1. Sumário estatístico
print(df.describe())
```

```
       consumo_potencia_kw  frequencia_atualizacao_s  energia_kwh  tempo_uso_h
count            60.000000                 60.000000    60.000000    60.000000
mean              1.252350                  8.750000     0.003492     0.002450
std               1.973011                  2.183334     0.005486     0.000611
min               0.001000                  5.000000     0.000000     0.001400
25%               0.077500                  8.750000     0.000225     0.002450
50%               0.275000                 10.000000     0.000800     0.002800
75%               1.400000                 10.000000     0.003900     0.002800
max               5.500000                 10.000000     0.015300     0.002800
```

3.2. Verificação de Valores Ausentes

Nesta subseção, verificamos a presença de valores ausentes nos dados de consumo de energia. A identificação de valores ausentes é crucial, pois pode impactar a análise e os resultados. A verificação é realizada utilizando o método `isna().sum()` do pandas, que retorna a contagem de valores ausentes para cada coluna do DataFrame.

```
# 2. Verificação de valores ausentes
print(df.isna().sum())
```

```
timestamp                 0
consumo_potencia_kw       0
frequencia_atualizacao_s  0
dispositivo               0
```

```
status                      0
energia_kwh                 0
tempo_uso_h                 0
dtype: int64
```

3.3. Identificação de Outliers

Nesta subseção, identificamos os outliers nos dados de consumo de energia. A identificação de outliers é importante para garantir que a análise não seja distorcida por valores atípicos. Utilizamos o método do intervalo interquartil (IQR) para identificar e remover outliers.

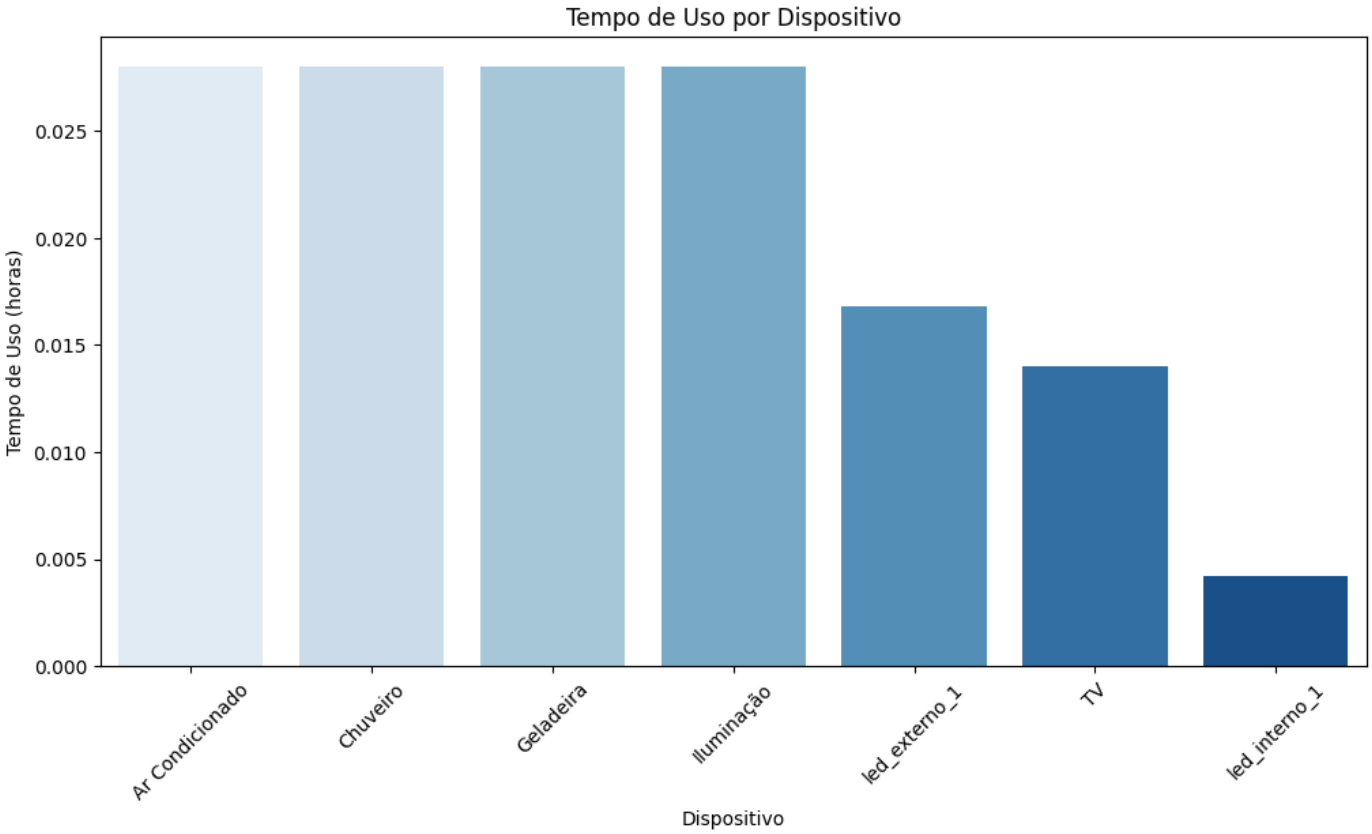
```
# 3. Análise de outliers (utilizando o método do intervalo interquartil - IQR) por dispositivo e status
def remove_outliers(group):
    q1 = group.quantile(0.25)
    q3 = group.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = group[(group < lower_bound) | (group > upper_bound)]
    print(f"Outliers removed for group {group.name}: {len(outliers)}")
    return group[(group >= lower_bound) & (group <= upper_bound)]


df["energia_kwh"] = df.groupby(["dispositivo", "status"])["energia_kwh"].apply(
    remove_outliers
).reset_index(level=[0, 1], drop=True)
```

```
Outliers removed for group ('Ar Condicionado', 'ON'): 0
Outliers removed for group ('Chuveiro', 'ON'): 0
Outliers removed for group ('Geladeira', 'ON'): 0
Outliers removed for group ('Iluminação', 'ON'): 0
Outliers removed for group ('TV', 'ON'): 0
Outliers removed for group ('led_externo_1', 'ligado_max'): 0
Outliers removed for group ('led_externo_1', 'ligado_min'): 0
Outliers removed for group ('led_interno_1', 'ligado'): 0
```

3.4. Gráfico de Barras de Tempo de Uso por Dispositivo

Nesta subseção, visualizamos o tempo total de uso por dispositivo utilizando um gráfico de barras. A visualização nos ajuda a entender quais dispositivos são utilizados por períodos mais longos, o que pode contribuir para o consumo elevado de energia.

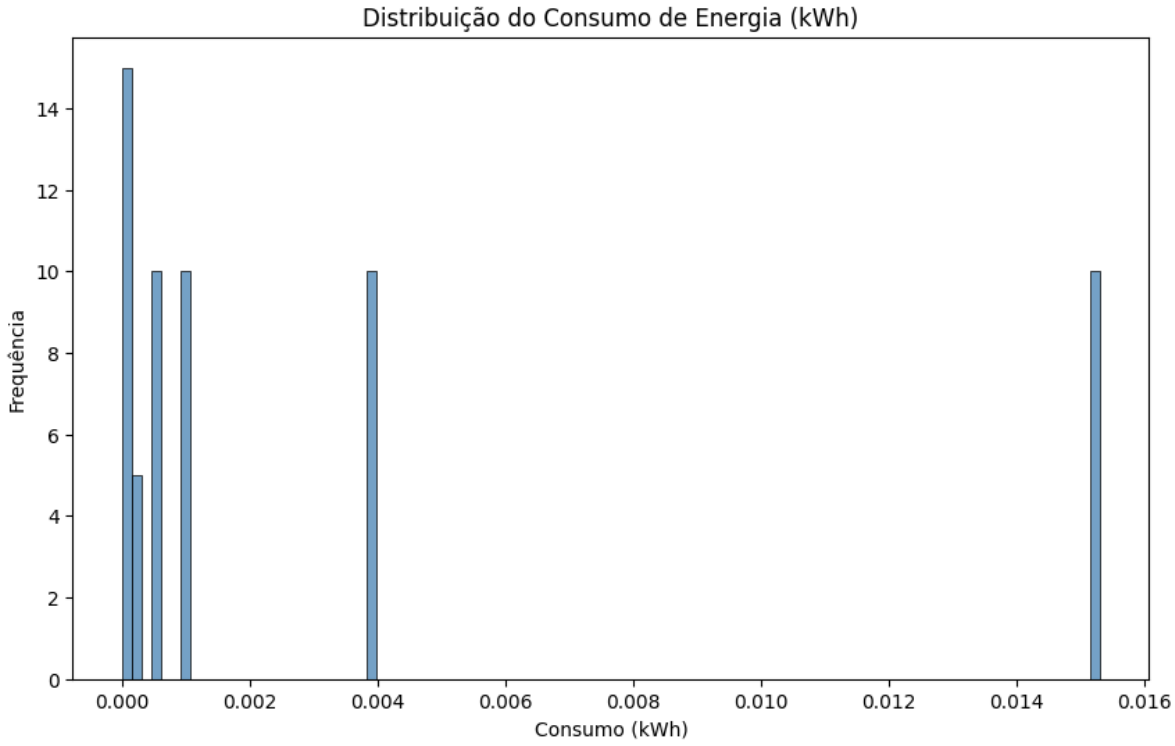
```
# 4. Gráfico de barras de tempo de uso por dispositivo

# Calcular o tempo total de uso por dispositivo
tempo_uso = df.groupby('dispositivo')['tempo_uso_h'].sum().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x=tempo_uso.index, y=tempo_uso.values, hue=tempo_uso.index, legend=False, palette='Blues')
plt.title('Tempo de Uso por Dispositivo')
plt.xlabel('Dispositivo')
plt.ylabel('Tempo de Uso (horas)')
plt.xticks(rotation=45)
plt.show()
```



3.5. Visualização da Distribuição do Consumo de Energia

Nesta subseção, visualizamos a distribuição do consumo de energia utilizando um histograma. A visualização nos ajuda a entender a frequência dos diferentes níveis de consumo de energia e identificar possíveis padrões ou anomalias.

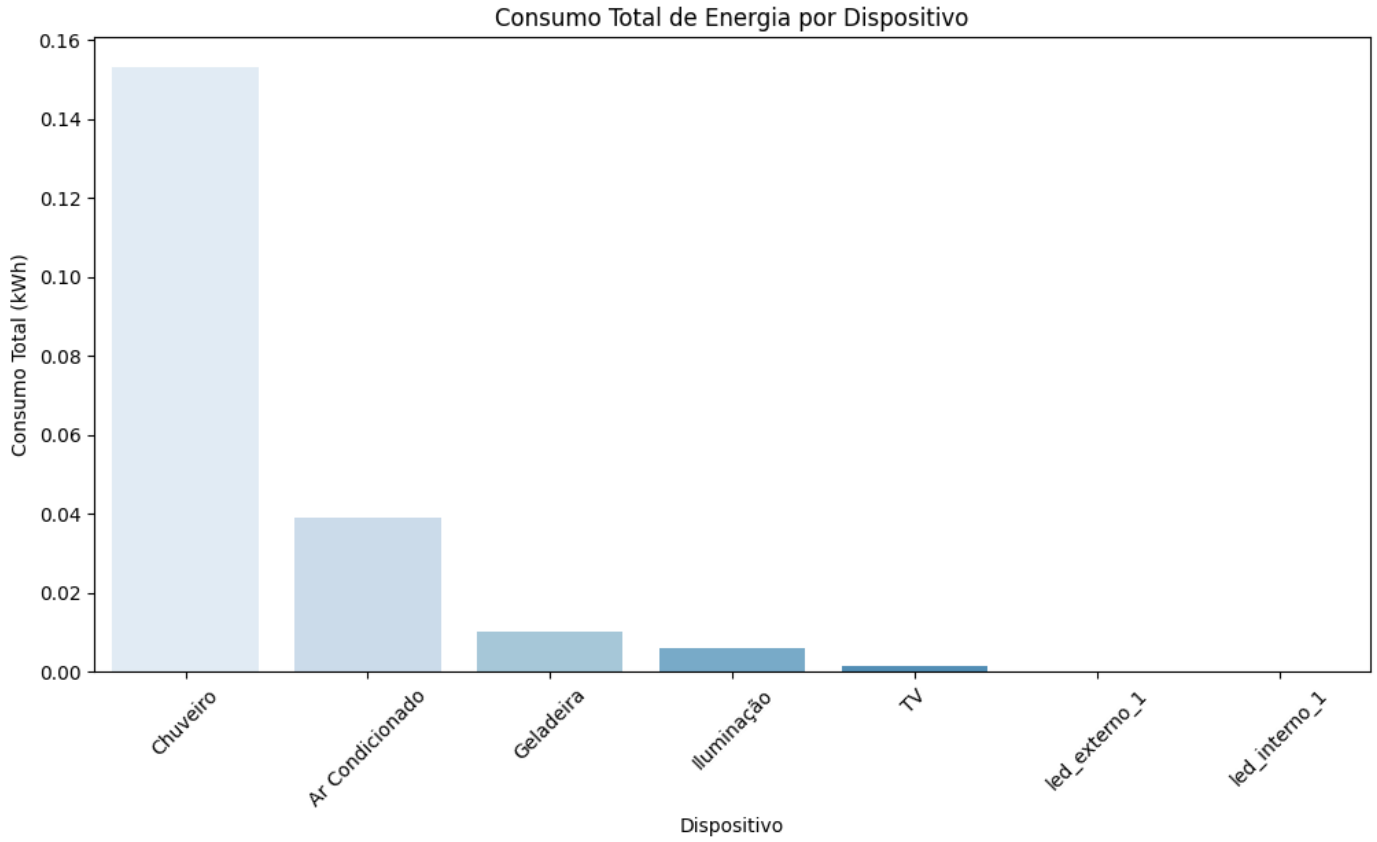
```
# 5. Histograma do consumo de energia
plt.figure(figsize=(10, 6))
sns.histplot(df['energia_kwh'], bins=100, kde=False, color='steelblue')
plt.title('Distribuição do Consumo de Energia (kWh)')
plt.xlabel('Consumo (kWh)')
plt.ylabel('Frequência')
plt.show()
```



3.6. Análise do Consumo de Energia por Dispositivo

Nesta subseção, analisamos o consumo de energia por dispositivo. A análise nos ajuda a identificar quais dispositivos consomem mais energia e a entender melhor os padrões de consumo de cada dispositivo.

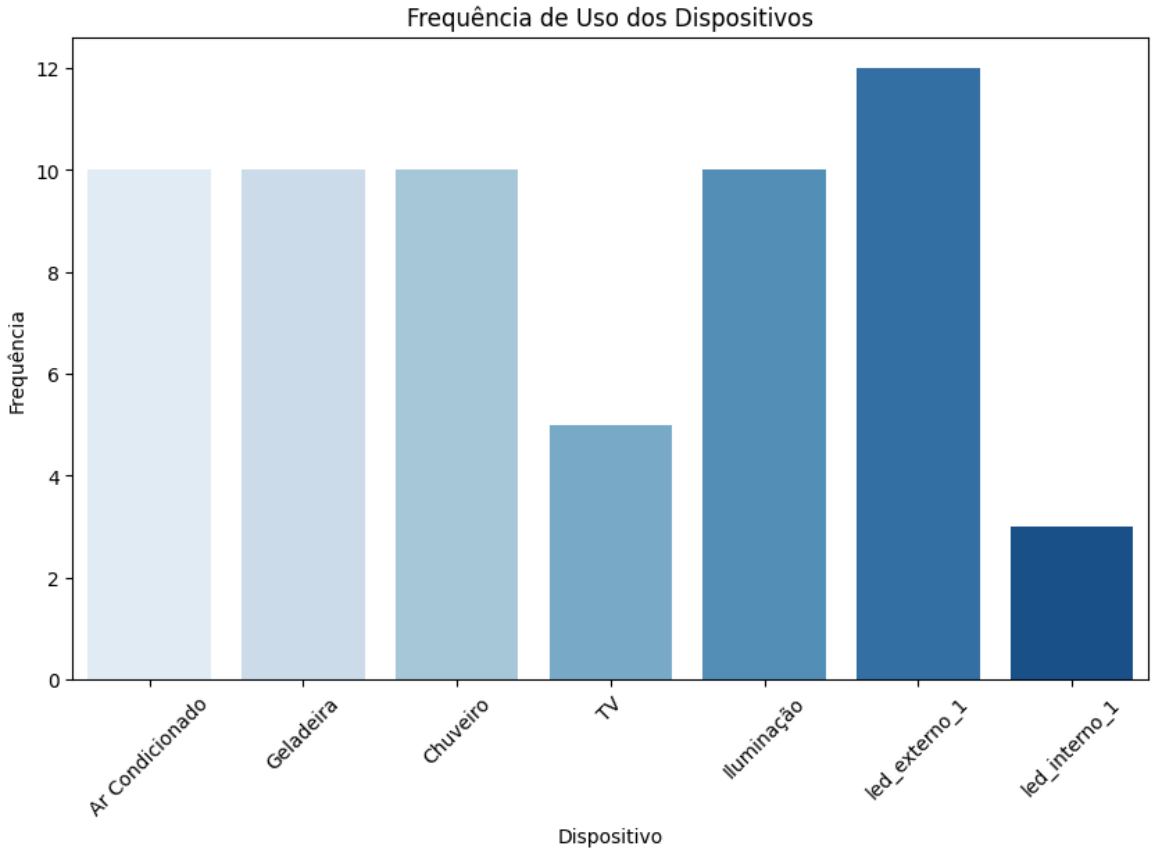
```
# 6. Gráfico de barras do consumo de energia por dispositivo
consumo_por_dispositivo = df.groupby('dispositivo')['energia_kwh'].sum().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x=consumo_por_dispositivo.index, y=consumo_por_dispositivo.values, hue=consumo_por_dispositivo.index,
legend=False, palette='Blues')
plt.title('Consumo Total de Energia por Dispositivo')
plt.xlabel('Dispositivo')
plt.ylabel('Consumo Total (kWh)')
plt.xticks(rotation=45)
plt.show()
```



3.7. Distribuição dos Dispositivos

Nesta subseção, visualizamos a distribuição dos dispositivos utilizando um gráfico de barras. A visualização nos ajuda a entender a frequência de uso dos diferentes dispositivos e identificar quais são mais utilizados.

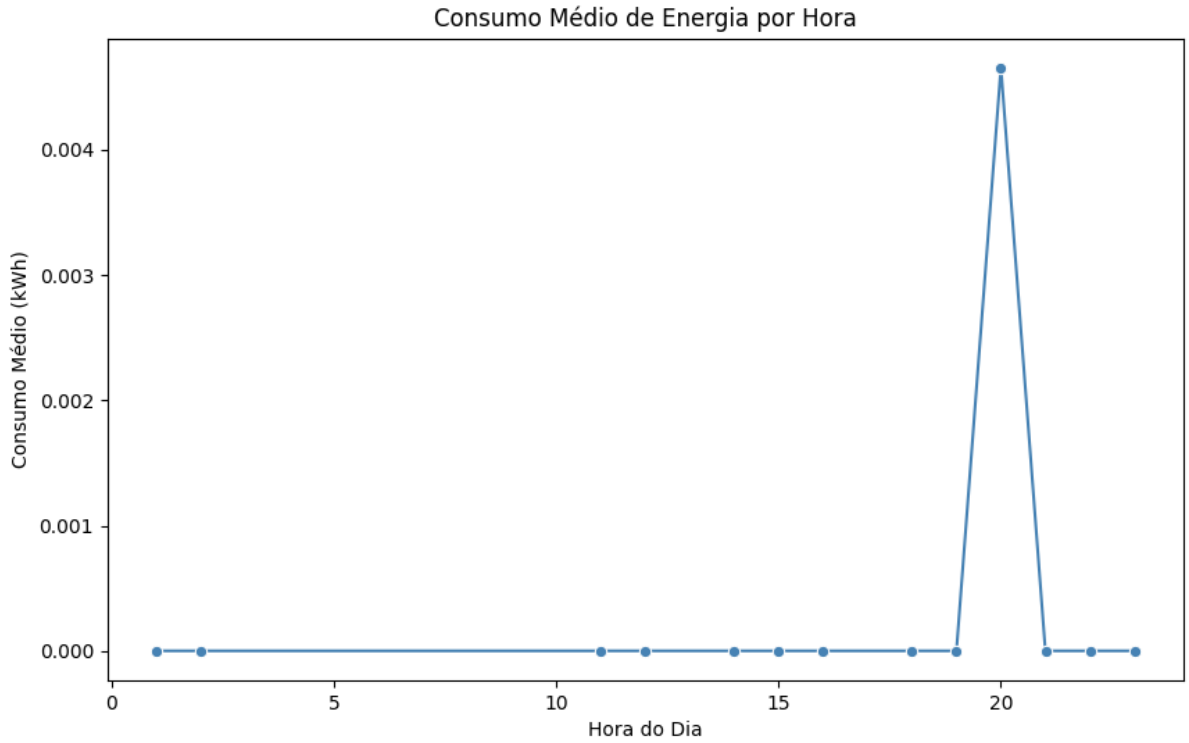
```
# 7. Distribuição dos dispositivos
plt.figure(figsize=(10, 6))
sns.countplot(x='dispositivo', data=df, hue='dispositivo', legend=False, palette='Blues')
plt.title('Frequência de Uso dos Dispositivos')
plt.xlabel('Dispositivo')
plt.ylabel('Frequência')
plt.xticks(rotation=45)
plt.show()
```



3.8. Análise da Relação entre Consumo e Tempo

Nesta subseção, analisamos a relação entre o consumo de energia e o tempo. A análise nos ajuda a entender como o consumo de energia varia ao longo do dia e identificar possíveis padrões de uso. Utilizamos um gráfico de linha para visualizar a média de consumo de energia por hora do dia.

```
# 8. Análise da relação entre consumo e tempo (utilize uma janela de tempo apropriada para melhor visualização)
df['hora'] = df['timestamp'].dt.hour
df_agregado = df.groupby('hora')['energia_kwh'].mean().reset_index()

plt.figure(figsize=(10, 6))
sns.lineplot(x='hora', y='energia_kwh', data=df_agregado, marker='o', color='steelblue')
plt.title('Consumo Médio de Energia por Hora')
plt.xlabel('Hora do Dia')
plt.ylabel('Consumo Médio (kWh)')
plt.show()
```



4. Conclusões

Nesta seção, apresentamos as principais conclusões obtidas a partir da análise exploratória de dados de consumo de energia residencial. As conclusões são baseadas nas visualizações e análises realizadas nas seções anteriores.

4.1. Padrões de Consumo de Energia

- **Distribuição do Consumo**: A distribuição do consumo de energia mostrou que a maioria dos registros de consumo está concentrada em valores mais baixos, com alguns outliers indicando consumos muito altos. Isso sugere que a maioria dos dispositivos consome pouca energia, mas alguns dispositivos específicos podem ter picos de consumo.
- **Consumo por Dispositivo**: A análise do consumo por dispositivo revelou que alguns dispositivos são responsáveis por uma parcela significativa do consumo total de energia. Identificar esses dispositivos pode ajudar a focar em estratégias de otimização de consumo.
- **Tempo de Uso por Dispositivo**: A análise do tempo de uso por dispositivo mostrou que alguns dispositivos são utilizados por períodos mais longos, o que pode contribuir para o consumo elevado de energia.

4.2. Outliers e Dados Faltantes

- **Identificação de Outliers**: A remoção de outliers utilizando o método do intervalo interquartil (IQR) ajudou a limpar os dados e garantir que a análise não fosse distorcida por valores atípicos. A presença de outliers pode indicar problemas de medição ou uso anômalo de dispositivos.
- **Dados Faltantes**: A verificação de valores ausentes mostrou que os dados estavam completos, não havendo lacunas. No entanto, é importante considerar técnicas de imputação de dados faltantes, como o uso do pacote `mice`, para garantir a integridade da análise caso lacunas estejam presentes.

4.3. Relação entre Consumo e Tempo

- **Consumo ao Longo do Dia**: A análise da relação entre consumo e tempo revelou padrões interessantes de uso ao longo do dia. O consumo de energia tende a variar em diferentes horas do dia, com picos em determinados períodos. Isso pode estar relacionado aos hábitos diários dos residentes, como horários de refeições e uso de eletrodomésticos.

4.4. Sugestões para Otimização do Consumo de Energia

- **Foco nos Dispositivos de Alto Consumo**: Dispositivos que consomem muita energia devem ser alvo de estratégias de otimização, como a substituição por modelos mais eficientes ou a implementação de práticas de uso consciente.
- **Monitoramento Contínuo**: Implementar sistemas de monitoramento contínuo pode ajudar a identificar padrões de consumo em tempo real e permitir ações corretivas imediatas.
- **Educação dos Residentes**: Educar os residentes sobre práticas de economia de energia pode contribuir significativamente para a redução do consumo. Informações sobre o impacto do uso de dispositivos específicos e dicas de economia podem ser úteis.

4.5. Novas Possibilidades de Melhoria para o Futuro do Sistema

- **Previsão de Consumo**: Implementar modelos de machine learning para prever o consumo de energia com base em dados históricos pode ajudar a identificar padrões e antecipar picos de consumo, permitindo uma melhor gestão da energia.

- **Automação de Ações**: Desenvolver sistemas de automação que ajustem automaticamente o uso de dispositivos com base em padrões de consumo e horários pode contribuir para a otimização do consumo de energia.
- **Análise de Custo-Benefício**: Realizar análises de custo-benefício para avaliar o impacto financeiro de diferentes estratégias de economia de energia e identificar as mais eficazes.
- **Feedback em Tempo Real**: Fornecer feedback em tempo real para os residentes sobre seu consumo de energia e dicas personalizadas para economizar pode incentivar comportamentos mais conscientes e reduzir o consumo.
- **Expansão do Sistema**: Expandir o sistema para incluir outros tipos de consumo, como água e gás, pode fornecer uma visão mais abrangente do uso de recursos e identificar oportunidades adicionais de economia.

4.6. Considerações Finais

A análise exploratória de dados de consumo de energia residencial forneceu insights valiosos sobre os padrões de uso e identificou áreas potenciais para otimização. A implementação de estratégias baseadas nessas conclusões pode contribuir para a redução do consumo de energia e, consequentemente, para a economia de custos e a sustentabilidade ambiental.

Agradecemos a oportunidade de realizar esta análise e esperamos que as conclusões apresentadas sejam úteis para a tomada de decisões informadas sobre o consumo de energia residencial.

Conclusão

Esta integração demonstra a capacidade de conectar diferentes tecnologias e criar um sistema completo para o gerenciamento de energia. A flexibilidade do sistema permite a expansão para incluir mais sensores, dispositivos e algoritmos de otimização avançados. A adição de dados reais e a análise estatística mais robusta em R, complementarão o projeto.