Escola Superior de Tecnologia e Gestão

**Smart Warehouse**

**Grupo 136 – EI PL**

João Parreira – 2221985

Francisco Francisco - 2221842

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, Junho de 2023

# Lista de Figuras

[Figura 1 - Arquitetura IoT genérica (altere para a sua) 2](#_Toc102681365)

[Figura 2 - Exemplo de um diagrama para representação de um evento 3](#_Toc102681366)

# Lista de tabelas

Elemento a figurar, **quando aplicável**.

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

# Lista de siglas e acrónimos

|  |  |
| --- | --- |
| ESTG | Escola Superior de Tecnologia e Gestão |
| IPLeiria  MCU  SBC | Instituto Politécnico de Leiria  Microcontroller Unit  Single Board Computer |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Cuidados na elaboração da lista de siglas e acrónimos:

* Ordenação alfabética;
* Apenas as que sejam relevantes para a leitura do texto.

Adicionar mais entradas à tabela, caso seja necessário (a tabela não tem contornos, mas está no texto).

Índice

[Lista de Figuras iv](#_Toc135735459)

[Lista de tabelas v](#_Toc135735460)

[Lista de siglas e acrónimos vi](#_Toc135735461)

[1. Introdução 1](#_Toc135735462)

[2. Arquitetura 2](#_Toc135735463)

[3. Implementação 3](#_Toc135735464)

[4. Cenário de Teste 4](#_Toc135735465)

[5. Resultados obtidos 5](#_Toc135735466)

[6. Conclusão 6](#_Toc135735467)

[7. Bibliografia 7](#_Toc135735468)

# Introdução

O presente relatório descreve o projeto de desenvolvimento de uma solução IoT (Internet das Coisas) para um Smart Warehouse, que tem como objetivo monitorar e controlar diversos aspectos do armazém de forma automatizada e inteligente. A solução utiliza tecnologias de Internet para integrar sensores, atuadores e uma dashboard de controle.

O tema abordado neste trabalho é a implementação de um sistema IoT para otimizar a gestão de um armazém, utilizando sensores de umidade, temperatura e luminosidade, bem como atuadores para controlar o ar condicionado, iluminação e portão. A solução visa melhorar a eficiência operacional do armazém, garantindo condições adequadas de temperatura e iluminação para os produtos armazenados, além de oferecer controle remoto e automação dos processos.

A relevância desse tema reside na crescente adoção de tecnologias IoT em diversos setores da indústria, incluindo a logística e o gerenciamento de armazéns. A implementação de um sistema inteligente nesse contexto traz benefícios como redução de custos operacionais, otimização de recursos, melhoria na qualidade dos produtos e aumento da segurança.

Os objetivos deste trabalho são:

1. Implementar um sistema IoT para monitorar e controlar o ambiente de um Smart Warehouse, considerando a temperatura, humidade, luminosidade e status dos atuadores (ar condicionado, iluminação e portão).
2. Desenvolver uma dashboard de controle que permita visualizar os dados dos sensores em tempo real, bem como controlar os atuadores de forma remota.
3. Realizar testes e avaliar os resultados obtidos com a implementação da solução IoT.
4. Analisar os benefícios e possíveis melhorias do sistema proposto.

Para atingir esses objetivos, foram utilizados métodos e técnicas baseados em tecnologias de Internet, como a integração de dispositivos IoT, o uso de APIs para comunicação entre os dispositivos e a dashboard, a programação de scripts em Python e C++, além da utilização de bibliotecas como a Chart.js para a criação de gráficos.

A estrutura deste relatório está organizada da seguinte maneira:

1. Introdução: apresentação do tema, justificativa, objetivos, métodos e estrutura do trabalho.
2. Arquitetura: descrição da arquitetura do sistema IoT, incluindo os dispositivos utilizados, a comunicação entre eles e a estrutura da dashboard.
3. Implementação: detalhes sobre a implementação dos componentes do sistema, incluindo o código Python e C++ utilizados nos dispositivos, a integração com a API e a criação da dashboard.
4. Cenário de Teste: descrição do cenário de teste utilizado para avaliar o funcionamento do sistema IoT.
5. Resultados Obtidos: apresentação dos resultados obtidos com a implementação da solução, incluindo gráficos, dados e análise dos resultados.
6. Conclusão: síntese dos principais pontos abordados no trabalho, discussão dos resultados, limitações e sugestões para trabalhos futuros.
7. Bibliografia: referências utilizadas como base teórica e fontes consultadas durante a realização do trabalho.

No próximo ponto, "Arquitetura", será descrita em detalhes a arquitetura do sistema IoT implementado, incluindo os dispositivos utilizados e a comunicação entre eles.

# Arquitetura

A solução desenvolvida para o Smart Warehouse baseia-se em uma arquitetura IoT (Internet das Coisas) que integra diversos dispositivos e componentes para monitorar e controlar o ambiente do armazém. A arquitetura geral da solução é representada na seguinte imagem:

[Inserir imagem representando a arquitetura da solução]

Figura 1 - Arquitetura IoT genérica (altere para a sua)

A seguir, descreveremos os principais componentes da arquitetura:

1. Sensores:
   1. Sensor de temperatura e humidade: Utilizado para monitorar as condições climáticas do armazém. Um sensor DHT (por exemplo, DHT11 ou DHT22) é conectado a um microcontrolador.
   2. Sensor de luminosidade (LDR): Responsável por medir a intensidade da luz ambiente no armazém. Um sensor LDR é conectado a outro microcontrolador.
2. Microcontroladores:
   1. Raspberry Pi: Um SBC (Single Board Computer) Raspberry Pi é utilizado para a leitura dos sensores de temperatura e umidade, além de simular o atuador de iluminação. Um script em Python é executado no Raspberry Pi para ler os valores dos sensores e controlar a iluminação simulada por meio de um LED. O Raspberry Pi também é responsável por fazer requisições POST e GET para a API.
   2. Arduino MKR1000: O Arduino MKR1000 é utilizado para a leitura do sensor LDR e para simular os atuadores do portão e ar condicionado. Um código em C++ é executado no Arduino para ler o valor do sensor LDR, transformá-lo em 1 ou 0 (ligado ou desligado), e fazer requisições POST e GET para a API.
3. API:
   1. Servidor na nuvem: A solução utiliza uma API hospedada em um servidor na nuvem. Essa API suporta requisições POST e GET, permitindo que os dispositivos enviem dados dos sensores e solicitem informações sobre os atuadores.
4. Dashboard:
   1. Interface de controle: A dashboard é acessada por meio de um software executado localmente em um computador pessoal. Através da dashboard, é possível visualizar as leituras dos sensores (temperatura, umidade, luminosidade), o estado dos atuadores (ar condicionado, iluminação, portão) e a última imagem capturada. A dashboard também oferece botões de controle para ligar/desligar o ar condicionado e a iluminação, além de abrir/fechar o portão.

É importante ressaltar que a arquitetura apresentada é uma representação genérica da solução proposta, sem especificar detalhes específicos dos dispositivos utilizados. A implementação real pode variar dependendo dos modelos e tecnologias escolhidas.

No próximo capítulo, "Implementação", serão abordados os detalhes específicos da implementação dos componentes da solução IoT, incluindo o código utilizado nos dispositivos, a integração com a API e a criação da dashboard.

# Implementação

A implementação do projeto para o Smart Warehouse envolve a integração de diferentes componentes e a criação de algoritmos para atender aos requisitos do sistema. Nesta seção, serão abordados os aspectos mais importantes da implementação, incluindo os algoritmos utilizados e os eventos definidos para o funcionamento do sistema.

Leitura dos sensores e controle dos atuadores:

Sensor de temperatura e humidade: Um sensor DHT conectado ao Raspberry Pi é utilizado para ler os valores de temperatura e umidade do ambiente do armazém. Um script em Python é executado no Raspberry Pi para obter esses valores e enviá-los para a API por meio de requisições POST.

Sensor de luminosidade (LDR): Um sensor LDR conectado ao Arduino MKR1000 é utilizado para medir a intensidade da luz ambiente. O código em C++ no Arduino lê o valor do sensor LDR e o envia para a API por meio de requisições POST.

Atuadores (ar condicionado, iluminação e portão): O Arduino MKR1000 também é responsável por simular os atuadores do ar condicionado, iluminação e portão. O código no Arduino faz requisições GET para obter o estado atual do ar condicionado, iluminação e portão da API. Com base nas leituras dos sensores (temperatura, luminosidade) e nos estados dos atuadores, o Arduino decide se os atuadores devem ser ligados ou desligados e envia as informações atualizadas para a API por meio de requisições POST.

Controle automático dos atuadores:

Ar condicionado: No modo automático, o ar condicionado é ativado se a temperatura medida for superior a 15°C ou inferior a 5°C. O código no Arduino verifica a temperatura obtida da API e liga ou desliga o ar condicionado com base nesse valor.

Iluminação: A iluminação é ligada se o valor do sensor LDR for maior ou igual a 512, indicando pouca luz natural. O código no Arduino verifica o valor do sensor LDR obtido da API e controla o estado da iluminação de acordo com esse valor.

Captura e upload da última imagem:

Uma simulação de webcam é utilizada para capturar a última imagem do armazém. Essa imagem é então enviada para a API por meio de uma requisição POST e armazenada para posterior visualização na dashboard.

Interface de controle na dashboard:

A dashboard apresenta a leitura dos sensores de umidade, temperatura e luminosidade, bem como o estado dos atuadores (ar condicionado, iluminação e portão) e a última imagem capturada. Além disso, a dashboard possui botões de controle para ligar/desligar o ar condicionado e a iluminação, e para abrir/fechar o portão.

O código na dashboard envia requisições POST para a API ao interagir com os botões de controle, atualizando o estado dos atuadores de acordo com as ações realizadas pelo usuário.

Exemplo de algoritmo de controle da iluminação:

Obter valor do sensor LDR da API (GET /ldr)

Se valor\_luminosidade >= 512:

Enviar estado da iluminação para a API (POST /iluminacao, valor: ligado)

Senão:

Enviar estado da iluminação para a API (POST /iluminacao, valor: desligado)

Exemplo de algoritmo de controle automático do ar condicionado:

Obter valor da temperatura da API (GET /temperatura)

Se valor\_temperatura > 15 ou valor\_temperatura < 5:

Enviar estado do ar condicionado para a API (POST /arcondicionado, valor: ligado)

Senão:

Enviar estado do ar condicionado para a API (POST /arcondicionado, valor: desligado)

Esses são exemplos simplificados dos algoritmos utilizados para controlar a iluminação e o ar condicionado com base nas leituras dos sensores e nos requisitos do projeto. Os demais eventos e algoritmos devem ser implementados de acordo com os requisitos específicos de cada componente e funcionalidade.

[Inserir Figura 2: Diagrama de fluxo representando as ações e eventos no sistema]

Figura 2 - Exemplo de um diagrama para representação de um evento

# Cenário de Teste

Para testar a solução IoT desenvolvida para o Smart Warehouse, criamos um cenário de teste que envolve a interação dos diferentes componentes e a verificação do correto funcionamento do sistema. Nesta seção, descreveremos em detalhes o cenário de teste, incluindo os softwares, procedimentos, equipamentos e suas configurações principais.

Softwares utilizados:

Sistema operacional: Raspberry Pi OS

Linguagem de programação: Python, C++

Bibliotecas: Flask (Python), Chart.js (JavaScript)

Ferramentas de simulação: Simulador de webcam (para captura de imagens)

Procedimentos realizados:

Instalação e configuração do Raspberry Pi:

Configuramos o Raspberry Pi com o sistema operacional adequado (Raspberry Pi OS) e realizamos as configurações iniciais, como conexão de rede e acesso SSH.

Instalamos as dependências necessárias, como o Flask, para executar o script Python responsável pela leitura dos sensores e interação com a API.

Conectamos o sensor DHT ao Raspberry Pi e fizemos as devidas conexões e configurações para garantir a leitura correta da temperatura e umidade.

Configuração do Arduino MKR1000:

Carregamos o código em C++ no Arduino MKR1000, que inclui a leitura do sensor LDR e a simulação dos atuadores (LEDs para iluminação e portão).

Realizamos as conexões necessárias, como o sensor LDR e os LEDs, de acordo com as especificações do projeto.

Configuração da API e armazenamento de dados:

Criamos a API utilizando o Flask no Raspberry Pi. Configuramos rotas para receber requisições POST e GET relacionadas aos sensores e atuadores.

Definimos os endpoints da API para receber os dados dos sensores (temperatura, umidade, luminosidade) e atuadores (ar condicionado, iluminação, portão).

Implementamos a funcionalidade de armazenamento dos dados em arquivos TXT, contendo nome, valor, data e log para cada sensor e atuador.

Desenvolvimento da dashboard e histórico:

Criamos a interface de dashboard utilizando HTML, CSS e JavaScript. Utilizamos a biblioteca Chart.js para gerar gráficos de histórico para temperatura e umidade.

Implementamos a página de histórico para as últimas 10 imagens capturadas, exibindo as imagens enviadas para a API por meio da simulação de webcam.

Equipamentos utilizados:

Raspberry Pi: Utilizamos um Single Board Computer (SBC) Raspberry Pi para executar o script Python, hospedar a API e gerar a dashboard.

Arduino MKR1000: Utilizamos o Arduino MKR1000 para ler o sensor LDR e simular os atuadores (iluminação e portão).

Sensor DHT: Utilizamos um sensor DHT conectado ao Raspberry Pi para obter as leituras de temperatura e umidade.

Sensor LDR: Utilizamos um sensor LDR conectado ao Arduino MKR1000 para medir a luminosidade ambiente.

Configurações principais:

Raspberry Pi:

Endereço IP: 192.168.0.100

Porta da API: 5000

Arduino MKR1000:

Porta de comunicação: Serial (padrão)

Configuração de conexão com a rede Wi-Fi

Imagem do cenário de teste:

[Inserir imagem descrevendo a disposição dos componentes no cenário de teste]

Com base nesse cenário de teste, verificamos a interação correta entre os componentes, incluindo a leitura dos sensores, controle dos atuadores, comunicação com a API e exibição dos dados na dashboard. Os resultados obtidos serão discutidos na próxima seção do relatório.

# Resultados obtidos

Neste ponto de apresentar todos os testes realizados para comprovarem se os objetivos foram alcançados. Podem utilizar tabelas, ou texto para descrever os vários testes realizados.

*Exemplo: no cenário, o script em Python do SBC está a realizar um pedido GET de 5 em 5 segundos à API, para obter o valor da temperatura. Quando o valor estiver acima de 20ºC, faz alterar soar um alerta sonoro durante 2 segundos e acende o LED de emergência.*

# Conclusão

Inserir aqui as conclusões.

Aqui, devem igualmente descrever e comentar em pormenor os resultados obtidos.

A conclusão:

* Deve ser sucinta;
* Não deve conter informações ou ideias novas;
* Análise da solução desenvolvida;
* Potencialidades e pontos a melhorar;
* Deve permitir concluir se se atingiram os objetivos enunciados na introdução.

Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão.

Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão.

# Bibliografia

Inserir aqui a bibliografia.

Bibliografia – quando se coloca toda a bibliografia consultada;