

# 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 do projeto .....	3
Figura 2 - Dashboard .....	5
Figura 3 – Pseudocódigo do algoritmo de iluminação .....	7
Figura 4 – Pseudocódigo do algoritmo do ar condicionado .....	8
Figura 5 – Evento de abertura e fecho do portão .....	8
Figura 6 –Esquema do ambiente de teste .....	11
Figura 6 –Ambiente de teste .....	11
Figura 7 – Gráfico do histórico da humidade utilizando a biblioteca chart.js .....	18
Figura 8 – Histórico das imagens capturadas .....	15

## Lista de siglas e acrónimos

API	Application Programming Interface
DHT	Digital Humidity and Temperature Sensor
IoT	Internet of Things
LDR	Light Dependent Resistor
LED	Light Emitting Diode
MCU	Microcontroller Unit
SBC	Single Board Computer

# Índice

Lista de Figuras .....	iv
Lista de siglas e acrónimos .....	v
1. Introdução .....	1
2. Arquitetura .....	3
3. Implementação.....	6
4. Cenário de Teste .....	9
5. Resultados obtidos .....	12
6. Conclusão .....	14
7. Bibliografia .....	16

# 1. Introdução

O presente relatório descreve o projeto de desenvolvimento de uma solução IoT (Internet das Coisas) para um Smart Warehouse, que tem como objetivo monitorizar e controlar diversos aspetos 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 humidade, 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 para os produtos armazenados e iluminação para os funcionários que aí trabalham, além de oferecer controle remoto e automação dos processos.

A relevância deste 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 monitorizar e controlar o ambiente de um Smart Warehouse, considerando a temperatura, humidade, luminosidade e estado 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.

## 2. Arquitetura

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

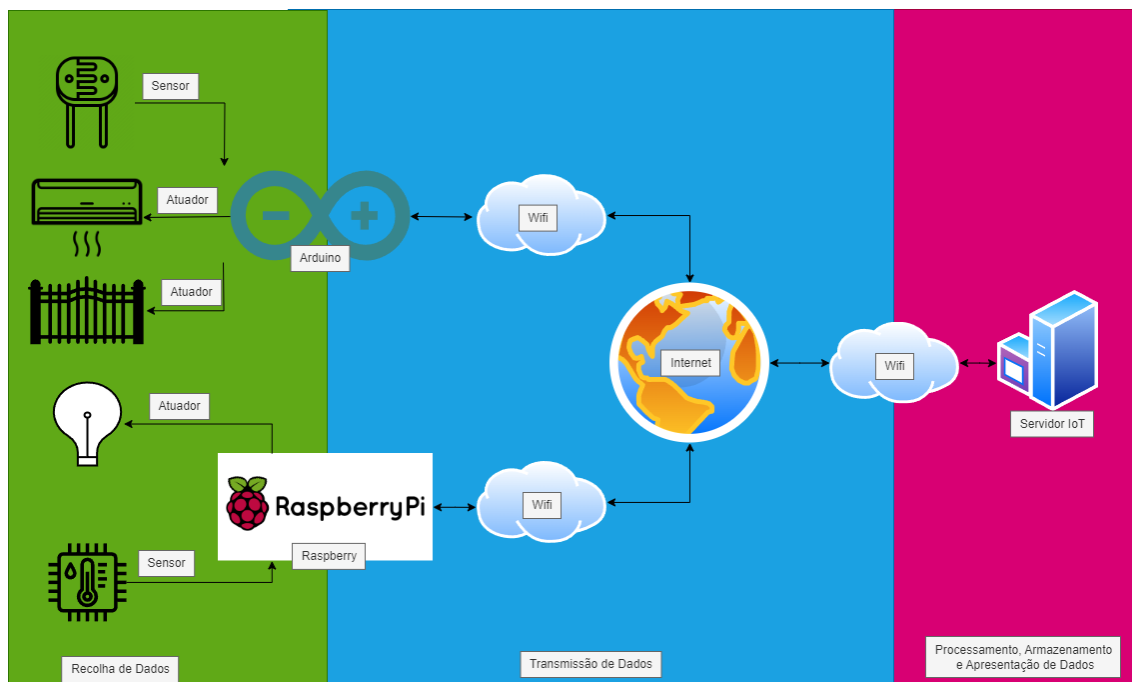


Figura 1 - Arquitetura IoT do projeto

A seguir, descreveremos os principais componentes da arquitetura:

### 1. Sensores:

- Sensor de temperatura e humidade:** Utilizado para monitorizar as condições climáticas do armazém. Um sensor DHT (por exemplo, DHT11 ou DHT22) é conectado a um microcontrolador.
- Sensor de luminosidade (LDR):** Responsável por medir a intensidade da luz ambiente no armazém. Um sensor LDR é conectado a outro microcontrolador.

### 2. Microcontroladores:

- Raspberry Pi:** Um SBC (Single Board Computer) Raspberry Pi é utilizado para a leitura dos sensores de temperatura e humidade, 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.

- b. Arduino MKR1000: O Arduino MKR1000 é utilizado para a leitura do sensor LDR e para simular os atuadores do portão e ar condicionado através de 2 LEDs. Um código em C++ é executado no Arduino para ler o valor do sensor LDR, transformá-lo em 1 ou 0 (Boa ou Má Luz Natural), e fazer requisições POST e GET para a API.

3. API:

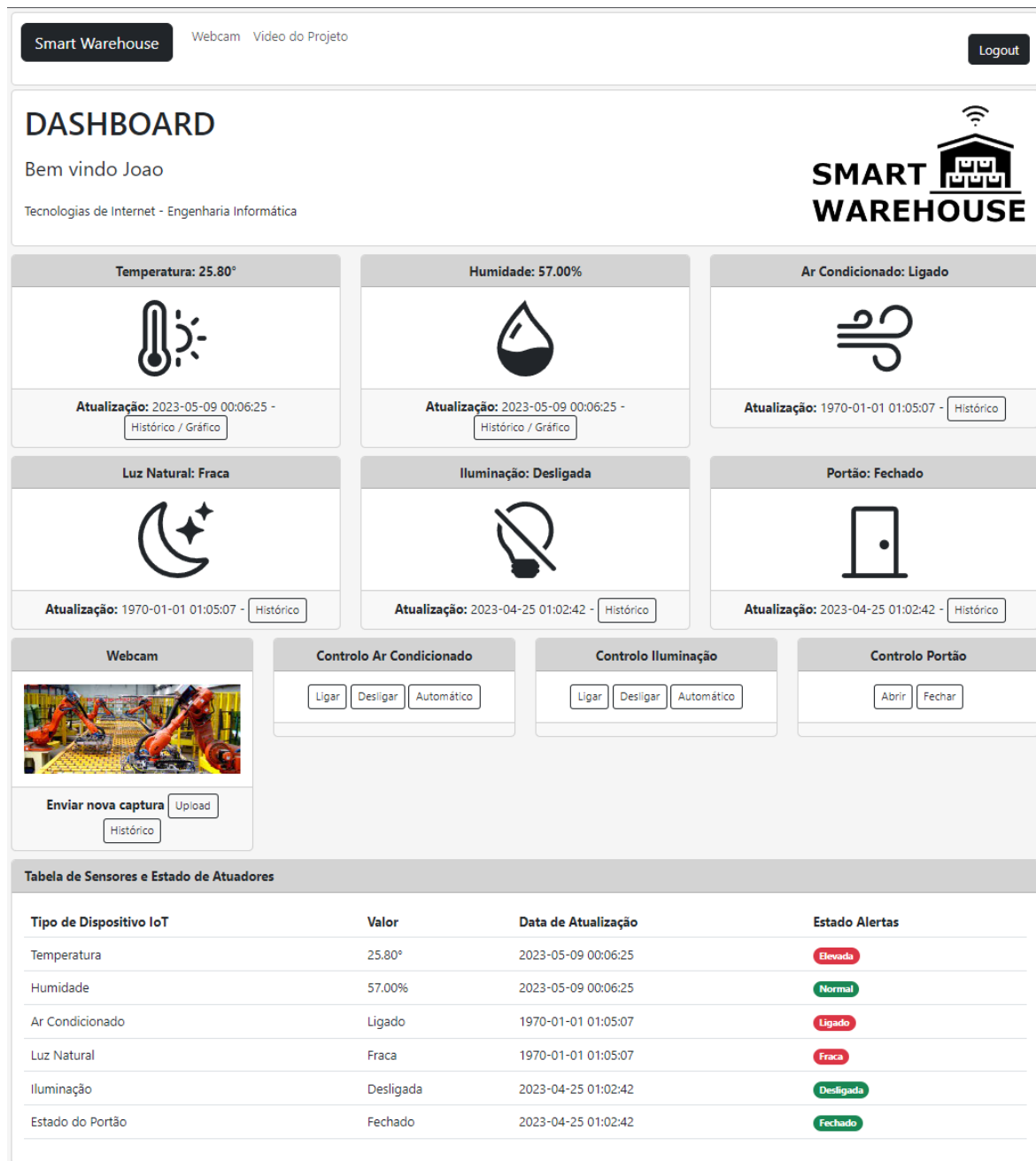
- a. Servidor na nuvem: A solução utiliza uma API hospedada em um servidor na nuvem. Essa API suporta requisições POST e GET, permitindo que os dispositivos enviem dados dos sensores e atuadores e solicitem informações sobre os valores dos sensores e estado dos atuadores.

4. Dashboard:

- a. Interface de controle: A dashboard é acessada por meio de um browser executado localmente em um computador pessoal. Através da dashboard, é possível visualizar as leituras dos sensores (temperatura, humidade, 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.

Figura 2 - Dashboard

### 3. 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.

1. Leitura dos sensores e controle dos atuadores:
  - a. Sensor de temperatura e humidade: Um sensor DHT conectado ao Raspberry Pi é utilizado para ler os valores de temperatura e humidade 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.
  - b. 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 envia para a API o valor 0 ou 1 (Fracas ou Boa Luz Natural) por meio de requisições POST.
  - c. Atuadores (ar condicionado, iluminação e portão): O Arduino MKR1000 também é responsável por simular os atuadores do ar condicionado e portão. O código no Arduino faz requisições GET para obter o estado atual do ar condicionado, temperatura e portão da API. Com base nas leituras dos sensores (temperatura) 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. Já o Raspberry é responsável por simular o atuador da iluminação. O código Python faz requisições GET para obter o estado atual da iluminação e luz natural da API. Com base nas leituras dos sensores (Luz Natural) e no estado do atuador, o Raspberry decide se o atuador deve ser ligado ou desligado e envia as informações atualizadas para a API por meio de requisições POST.
2. Controle automático dos atuadores:
  - a. 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.
  - b. Iluminação: A iluminação é ligada se o valor do sensor LDR for maior ou igual a 512, indicando pouca luz natural. O código no Raspberry Pi verifica o valor da

Luz Natural obtido da API e controla o estado da iluminação de acordo com esse valor.

3. Captura e upload da última imagem:
  - a. 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. É verificada a extensão e o tamanho do ficheiro antes de o gravar com o formato data.extensão. No máximo são armazenadas 10 imagens sendo a mais antiga descartada.
4. Interface de controle na dashboard:
  - a. A dashboard apresenta a leitura dos sensores de humidade, 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.
  - b. 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 da luz natural da API (GET /luz)
Obter valor da iluminação da API (GET /iluminação)
Se valor iluminação == 0 || Valor iluminação == 1
    Se valor_luz == 0:
        Ligar Iluminação
        Enviar estado da iluminação para a API (POST /iluminacao, valor: ligado)
    Se valor_luz == 1:
        Desligar Iluminação
        Enviar estado da iluminação para a API (POST /iluminacao, valor: desligado)
Se valor iluminação == 2
    Desligar Iluminação
Se valor iluminação == 3
    Ligar Iluminação
```

Figura 3 – Pseudocódigo do algoritmo de iluminação

Exemplo de algoritmo de controle automático do ar condicionado:

```
Obter valor da temperatura da API (GET /temperatura)
Obter valor do ar condicionado da API (GET /ac)
Se valor ac == 0 || Valor ac == 1
    Se valor_temperatura > 15 ou valor_temperatura < 5:
        Ligar Ar Condicionado
        Enviar estado do ar condicionado para a API (POST /ac, valor: ligado)
    Senão:
        Desligar Ar Condicionado
        Enviar estado da iluminação para a API (POST /ac, valor: desligado)
Se valor iluminação == 2
    Desligar Ar Condicionado
Se valor iluminação == 3
    Ligar Ar Condicionado
```

Figura 4 – Pseudocódigo do algoritmo do ar condicionado

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.

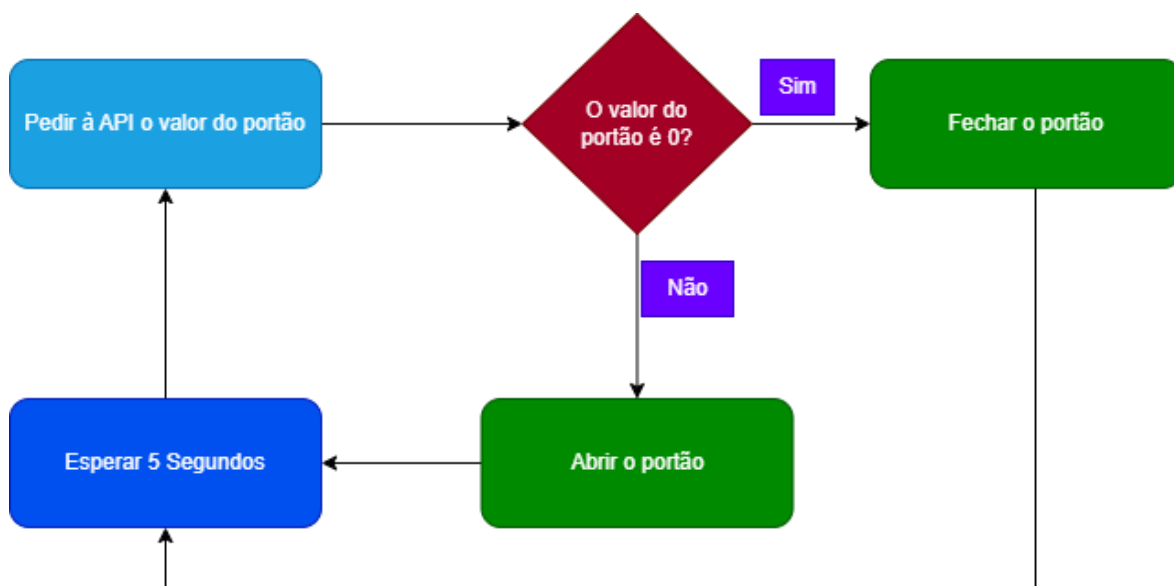


Figura 5 – Evento de abertura e fecho do portão

## 4. 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.

1. Softwares utilizados:
  - a. Sistema Operativo: Kernel Linux disponibilizado na UC
  - b. Linguagem de programação: Python, C++, JavaScript, PHP.
  - c. Bibliotecas: AdafruitDHT, cv2, requests, time, math, datetime (Python), Chart.js (JavaScript), Bootstrap (CSS / JavaScript),
  - d. Ferramentas de simulação: Simulador de webcam (para captura de imagens)
2. Procedimentos realizados:
  - a. Instalação e configuração do Raspberry Pi:
    - i. Configurámos o Raspberry Pi com o sistema operacional adequado (Linux Kernel) e realizamos as configurações iniciais, como conexão de rede e acesso SSH.
    - ii. Instalámos as dependências necessárias, como o AdafruitDHT e RPi, para executar o script Python responsável pela leitura dos sensores e interação com a API.
    - iii. Conectámos o sensor DHT ao Raspberry Pi e fizemos as devidas conexões e configurações para garantir a leitura correta da temperatura e humidade.
  - b. Configuração do Arduino MKR1000:
    - i. Carregámos o código em C++ no Arduino MKR1000, que inclui a leitura do sensor LDR e a simulação dos atuadores (LEDs para iluminação e portão).
    - ii. Realizámos as conexões necessárias, como o sensor LDR e os LEDs, de acordo com as especificações do projeto.
  - c. Configuração da API e armazenamento de dados:
    - i. Criámos a API utilizando PHP no servidor Cloud. Configurámos as rotas para receber requisições POST e GET relacionadas aos sensores e atuadores.
    - ii. Definimos os endpoints da API para receber os dados dos sensores (temperatura, humidade, luminosidade) e atuadores (ar condicionado, iluminação, portão).

- iii. Implementamos a funcionalidade de armazenamento dos dados em arquivos TXT, contendo nome, valor, data e log para cada sensor e atuador.
  - d. Desenvolvimento da dashboard e histórico:
    - i. Criamos a interface de dashboard utilizando HTML, CSS e JavaScript. Utilizamos a biblioteca Chart.js para gerar gráficos de histórico para temperatura e humidade.
    - ii. Implementamos a página de histórico para as últimas 10 imagens capturadas, exibindo as imagens enviadas para a API por meio da simulação de webcam.
3. Equipamentos utilizados:
- a. Raspberry Pi: Utilizamos um Single Board Computer (SBC) Raspberry Pi para executar o script Python.
  - b. Arduino MKR1000: Utilizamos o Arduino MKR1000 para ler o sensor LDR e simular os atuadores (iluminação e portão).
  - c. Computador Pessoal: Utilizamos um laptop para simular um servidor utilizando o software Uniform Server, para hospedar a API e gerar a dashboard.
  - d. Sensor DHT: Utilizamos um sensor DHT conectado ao Raspberry Pi para obter as leituras de temperatura e umidade.
  - e. Sensor LDR: Utilizamos um sensor LDR conectado ao Arduino MKR1000 para medir a luminosidade ambiente.
  - f. LEDs: Utilizamos 3 LEDs para simular os atuadores.
4. Configurações principais:
- a. Raspberry Pi:
    - i. Configuração de conexão com a rede Wi-Fi
    - ii. Ligação à API através do IP do laptop
  - b. Arduino MKR1000:
    - i. Configuração de conexão com a rede Wi-Fi
    - ii. Ligação à API através do IP do laptop
  - c. Laptop:
    - i. Configuração de conexão com a rede Wi-Fi
    - ii. Execução do Uniform Server para simular um servidor web.

## 5. Imagem do cenário de teste:

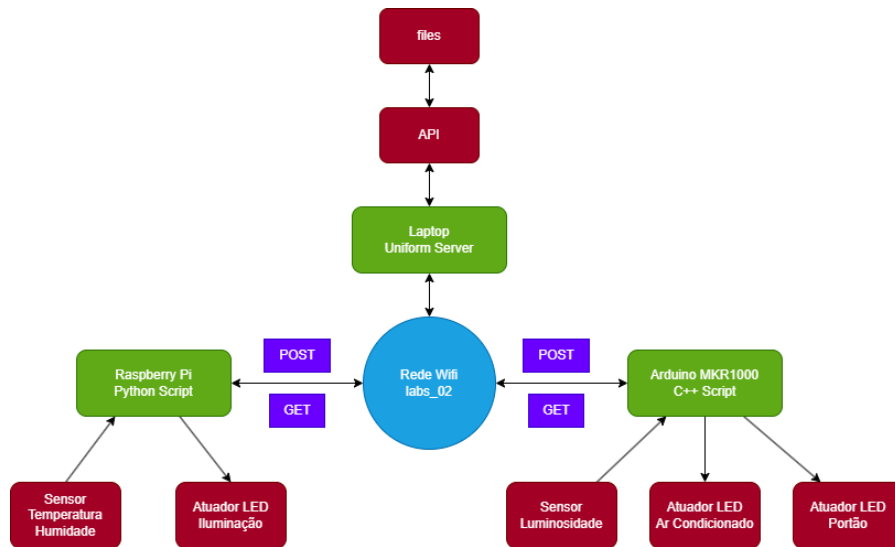


Figura 6 – Esquema do ambiente de teste

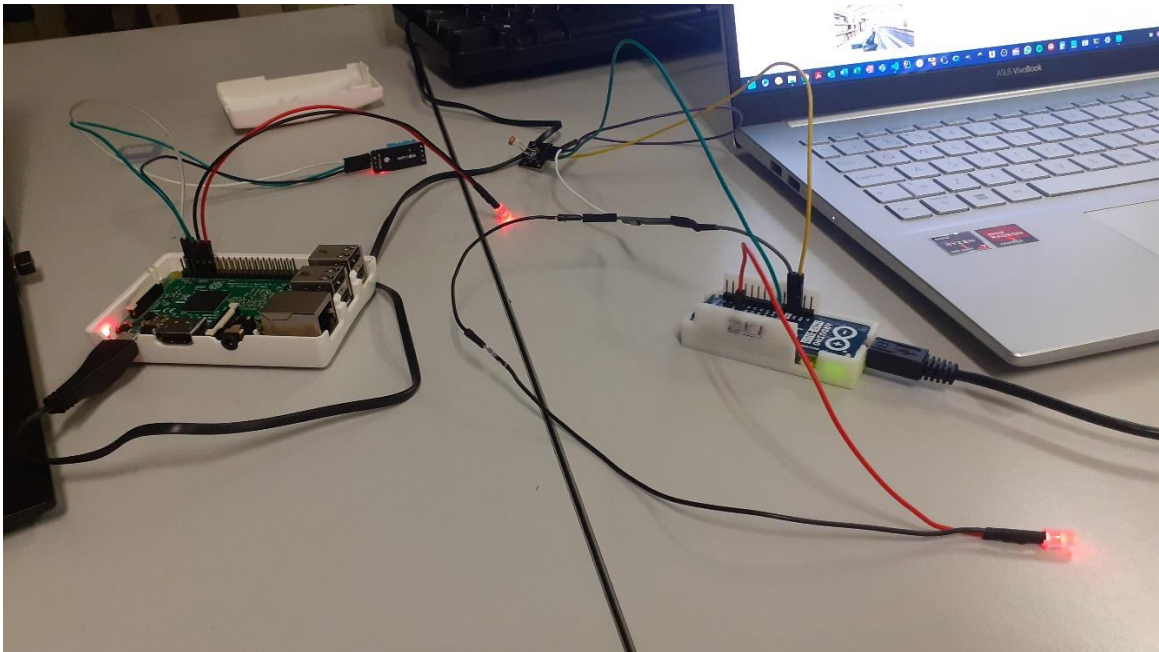


Figura 7 – Ambiente de teste

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.





## 5. Resultados obtidos

Durante o desenvolvimento e teste da solução IoT para o Smart Warehouse, foram realizados diversos testes para verificar se os objetivos propostos foram alcançados. Abaixo, apresentamos alguns dos principais testes realizados e os resultados obtidos:

1. Teste de leitura dos sensores e atuadores:
  - a. O script Python executado no Raspberry Pi obteve com sucesso as leituras dos sensores de temperatura e humidade.
  - b. O LED utilizado para simular a iluminação foi controlado corretamente de acordo com os valores lidos pelo sensor LDR.
  - c. O Arduino MKR1000, configurado com um sensor LDR e com LEDs simulando o portão e o ar condicionado, respondeu adequadamente às leituras e alterações de estado.
2. Teste de interação com a API:
  - a. A API desenvolvida suportou corretamente as requisições POST e GET realizadas pelos scripts Python e Arduino.
  - b. Os dados dos sensores (temperatura, humidade, luminosidade) e atuadores (ar condicionado, iluminação, portão) foram corretamente armazenados em arquivos TXT.
  - c. As páginas de histórico da dashboard exibiram corretamente os dados registados nos arquivos TXT.
  - d. A camera capturou as imagens sempre que a luz está acesa.
3. Teste de controle dos atuadores:
  - a. Os botões de controle na dashboard permitiram ligar e desligar o ar condicionado e a iluminação manualmente, conforme esperado.
  - b. No modo automático, o ar condicionado ligou quando a temperatura estava acima de 15°C ou abaixo de 5°C, e desligou quando a temperatura se encontrou dentro desses limites.
  - c. A iluminação foi ligada quando a luminosidade medida pelo sensor LDR foi igual ou superior a 512, e desligada quando a luminosidade foi inferior a 512. O modo manual de ligar/desligar também funcionou corretamente.
4. Teste de captura e exibição de imagens:
  - a. O simulador de webcam permitiu a captura de imagens e o envio para a API.
  - b. A página de histórico das últimas 10 imagens na dashboard exibiu corretamente as imagens capturadas.

Em geral, os testes realizados comprovaram que a solução IoT para o Smart Warehouse atingiu seus objetivos. A integração entre os componentes, a leitura dos

sensores, o controle dos atuadores e o armazenamento de dados foram bem-sucedidos. A dashboard apresentou corretamente as informações em tempo real, permitindo a monitorização e controle do ambiente do armazém de forma eficiente.

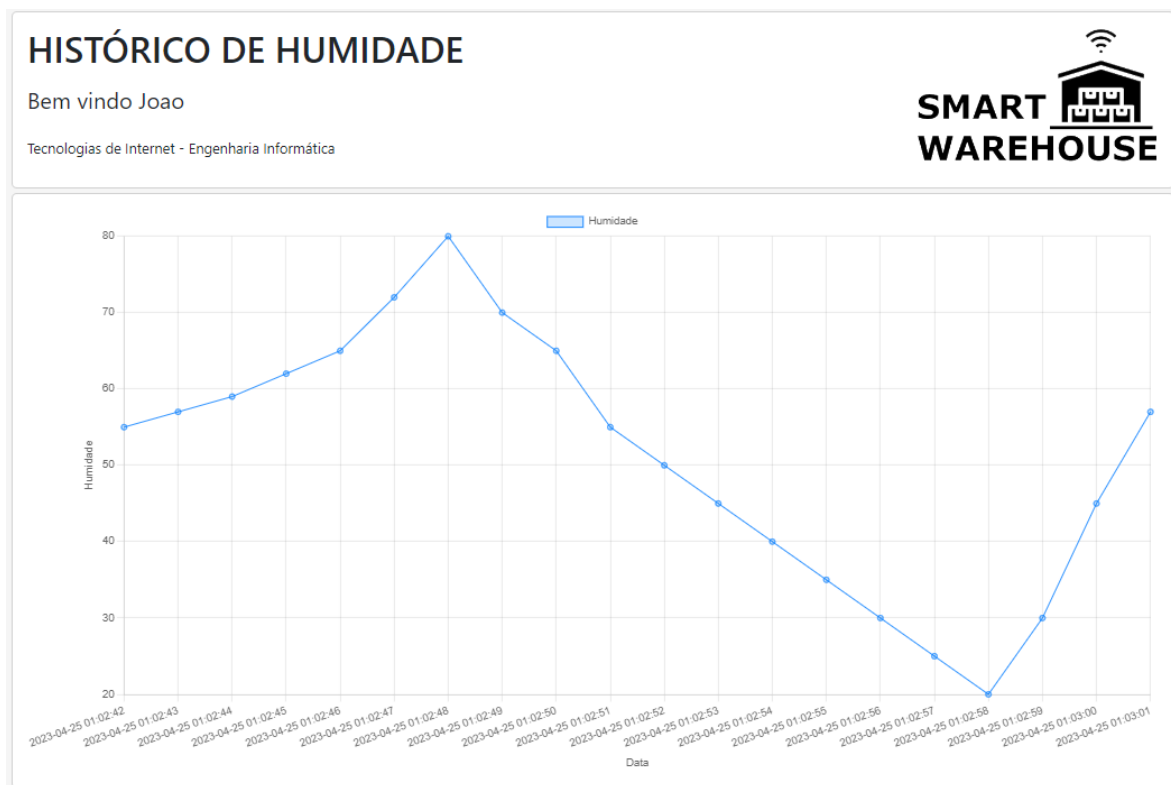


Figura 8 – Gráfico do histórico da humidade utilizando a biblioteca chart.js

## 6. Conclusão

A solução desenvolvida para o Smart Warehouse, baseada na tecnologia IoT, proporcionou um ambiente de armazenamento inteligente e eficiente, com controle e monitorização dos sensores e atuadores por meio de uma dashboard intuitiva. A partir dos resultados obtidos, é possível tirar as seguintes conclusões:

A arquitetura implementada, utilizando o Raspberry Pi e o Arduino MKR1000, mostrou-se adequada para a coleta de dados dos sensores de temperatura, humidade e luminosidade, bem como para o controle dos atuadores de ar condicionado, iluminação e portão. Os scripts em Python e C++ executados nos dispositivos desempenharam corretamente suas funções, permitindo a interação com a API e o envio de dados para a dashboard.

A dashboard desenvolvida apresentou as informações de forma clara e organizada, permitindo o acompanhamento em tempo real dos dados dos sensores, estado dos atuadores e visualização da última imagem capturada pela simulação da webcam. Além disso, as páginas de histórico, com gráficos para temperatura e humidade, proporcionaram uma análise mais aprofundada das variações ao longo do tempo.

A solução demonstrou ser capaz de atender aos objetivos propostos no projeto, como o controle automático do ar condicionado e da iluminação com base nas leituras dos sensores, a captura e exibição de imagens, e o armazenamento adequado dos dados em arquivos TXT.

Embora a solução tenha alcançado os objetivos estabelecidos, algumas melhorias podem ser consideradas. Por exemplo, a implementação de recursos adicionais, como alertas por e-mail ou integração com outros sistemas de gestão de armazéns, poderia agregar valor ao projeto. Além disso, a segurança da solução e a proteção dos dados devem ser consideradas em futuras iterações, utilizando práticas como criptografia e autenticação adequadas.

No geral, a solução IoT desenvolvida para o Smart Warehouse demonstrou seu potencial para otimizar a gestão do armazém, fornecendo informações em tempo real e permitindo o controle eficiente dos atuadores com base nas condições ambientais. O projeto serviu como uma experiência prática para a aplicação de tecnologias de Internet e Internet das Coisas em um cenário real, contribuindo para a compreensão e aprimoramento desses conceitos.

Em conclusão, o projeto atingiu seus objetivos ao criar uma solução IoT funcional e adequada para o gerenciamento de um Smart Warehouse, oferecendo um ambiente inteligente, eficiente e automatizado para o armazenamento de produtos.

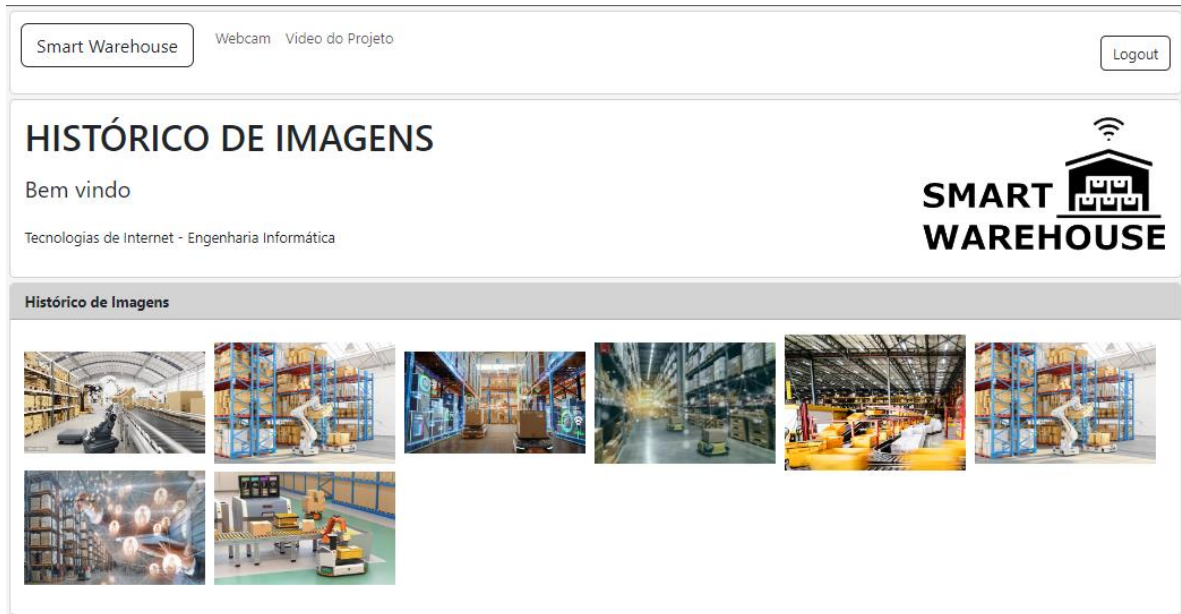


Figura 9 – Histórico de imagens capturadas

## 7. Bibliografia

1. Slides Teóricos da Unidade Curricular de Tecnologias de Internet da Escola Superior de Tecnologia e Gestão
2. Laboratórios Práticos da Unidade Curricular de Tecnologias de Internet da Escola Superior de Tecnologia e Gestão
3. W3Schools. (2021). PHP Tutorial. Retrieved from <https://www.w3schools.com/php/>.
4. Bootstrap. Retrieved from <https://getbootstrap.com/docs/5.3/getting-started/introduction/>.
5. Google – Search engine for solutions <https://www.google.com/>
6. Chat GPT – Retrieved from <https://openai.com/blog/chatgpt>
7. Raspberry Pi Foundation. (2021). Raspberry Pi. Retrieved from <https://www.raspberrypi.org/>
8. Arduino. (2021). Arduino. Retrieved from <https://www.arduino.cc/>
9. W3Schools. (2021). JavaScript Charts & Graphs. Retrieved from [https://www.w3schools.com/js/js\\_charting.asp](https://www.w3schools.com/js/js_charting.asp)
10. <https://requests.readthedocs.io/en/latest/user/quickstart/#post-a-multipart-encoded-file>