
# Produce Shop

Duarte de Abreu Sousa Lacerda - 2242974

Rafael Maria Coutinho de Freitas - 2230810

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, junho de 2025

# Lista de Figuras

Elemento a figurar, **quando aplicável**.

# Lista de siglas e acrónimos

Elemento a figurar, **quando aplicável**.

| | |
|---|---|
| ESTG | Escola Superior de Tecnologia e Gestão |
| IPLeiria | Instituto Politécnico de Leiria |
| IoT | Internet of Things |
| TI | Tecnologias de Internet |

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

# 1. Introdução

O presente relatório documenta o desenvolvimento de um projeto prático no âmbito da unidade curricular de Tecnologias para a Internet, com o objetivo de criar uma solução IoT (Internet of Things) para um supermercado inteligente. A solução permite monitorizar e controlar remotamente diversos sensores e atuadores, promovendo a automação e eficiência de operações no espaço físico simulado.

A pertinência do tema está associada à crescente aplicação de tecnologias IoT em ambientes comerciais, onde a recolha e análise de dados em tempo real pode ajudar a reduzir custos, melhorar a experiência do cliente e otimizar recursos, como ventilação ou deteção de obstáculos em zonas de acesso reservado.

Objetivos do trabalho:

Geral: Desenvolver um protótipo funcional de um sistema IoT para supervisão e controlo de um supermercado.

Específicos:

Implementar sensores para recolha de dados ambientais e de movimento;

Desenvolver uma API para gerir e disponibilizar os dados dos sensores;

Criar um painel web com funcionalidades de visualização e histórico;

Automatizar ações com base em condições definidas (ex: ligar um led quando a temperatura estiver elevada).

Métodos e técnicas utilizados:

Foram utilizados sensores físicos ligados a um Raspberry Pi 3 Model B, com scripts Python para recolha e registo de dados. A interface web foi desenvolvida com HTML, CSS, PHP e Bootstrap. O armazenamento dos dados é feito em ficheiros .txt, sem recurso a base de dados, conforme os requisitos. O projeto recorreu ainda a autenticação de utilizador e uma API local que disponibiliza os dados para o painel.

Estrutura do relatório:

Capítulo 2 – Arquitetura: descrição da arquitetura geral do sistema IoT proposto;

Capítulo 3 – Implementação: explicação dos principais algoritmos e eventos desenvolvidos;

Capítulo 4 – Cenário de Teste: descrição técnica do ambiente de testes, configurações e ferramentas utilizadas;

Capítulo 5 – Resultados obtidos: apresentação e análise dos testes realizados;

Capítulo 6 – Conclusão: reflexão sobre os resultados e considerações finais;

Capítulo 7 – Bibliografia: referências utilizadas no decorrer do projeto.

# 2. Arquitetura

A arquitetura da solução proposta para o projeto "Supermercado Inteligente" baseia-se num sistema IoT distribuído, composto por sensores e atuadores ligados a um microcontrolador central (Raspberry Pi 3 Model B), que comunica com uma API local responsável pela gestão e disponibilização dos dados recolhidos. A informação processada é posteriormente apresentada numa interface web desenvolvida com tecnologias de front-end e back-end como HTML, CSS, PHP e Bootstrap.

Esta arquitetura simula um sistema real aplicado num supermercado, permitindo monitorizar condições ambientais e realizar ações automatizadas com base em eventos definidos. A figura abaixo representa a arquitetura geral da solução.

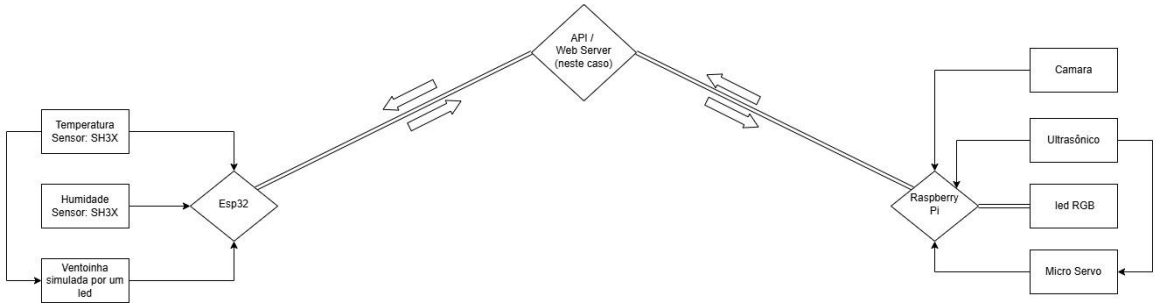


**Figura 1 – Arquitetura do projeto**

**Componentes principais:**

- **Sensores:**
    - Sensor de **temperatura** – monitoriza a temperatura ambiente;
    - Sensor de **humidade** – mede os níveis de humidade do ar;
    - Sensor **ultrassónico** – deteta obstáculos ou presença numa zona (ex: entrada de armazém).
- **Atuadores:**
    - **Ventoinha** – simulada por um led que liga quando uma certa temperatura for detetada.
    - **Servo motor** – usado para simular abertura/fecho de uma cancela.

- **LED RGB** – altera a sua cor entre verde, amarelo e vermelho, dependendo da quantidade de humidade detetada.

- **Microcontrolador / Unidade Central:**
  - **Raspberry Pi 3 Model B** – responsável por recolher dados de sensores, enviá-los para a API e interagir com os atuadores.
  - **Esp32 Dev Kit** – responsável por recolher dados de sensores, enviá-los para a API e interagir com os atuadores.

- **API local:**
  - Desenvolvida em PHP, esta API permite que a interface web obtenha dados e os apresente ao utilizador autenticado. Também suporta ações como histórico e estado atual dos sensores.

- **Interface Web:**
  - Painel responsivo para consulta de dados em tempo real, histórico, gráficos e ações. Desenvolvido com HTML, CSS, PHP e Bootstrap, sem frameworks adicionais.

- **Armazenamento:**
  - Cada sensor guarda os dados em ficheiros .txt locais no formato:
    - valor.txt, hora.txt, nome.txt e log.txt (formato dos logs: Y/M/D h:m:s;valor)

# 3. Implementação

A implementação do projeto do supermercado inteligente foi realizada em várias fases, desde a configuração dos sensores e atuadores no Raspberry Pi 3 Model B até ao desenvolvimento da interface web e da API local em PHP.

**Sensores e Registo de Dados**

Cada sensor está associado a um conjunto de ficheiros .txt onde são guardadas as leituras, nomeadamente:

valor.txt – último valor registado;

hora.txt – hora da última leitura;

nome.txt – nome do sensor/atuador;

log.txt – histórico de leituras no formato YYYY/MM/DD hh:mm:ss;valor.

A recolha dos dados foi feita através de scripts em Python executados periodicamente (via cron ou loop infinito), e os valores foram atualizados automaticamente nestes ficheiros.

**Atuadores e Ações Automatizadas**

Os atuadores foram configurados para reagir a eventos com base nas leituras dos sensores. A lógica de decisão foi implementada diretamente nos scripts Python. Um exemplo de evento implementado:

Evento 1: Quando a temperatura ultrapassa os 25 °C, a ventoinha é ativada durante 10 segundos e um LED vermelho é aceso.

Código simplificado (Python):

```
if temperatura >= 25:
    ligar_ventoinha()
    acender_led_vermelho()
elif temperatura < 25:
    desligar_ventoinha()
    apagar_led_vermelho()
```

**Outro exemplo prático:**

Evento 2: Quando o sensor ultrassónico deteta algo a menos de 10 cm, o servo motor abre uma porta simulada durante cerca de 5 segundos.

Estes eventos foram registados no log.txt de cada sensor/atuador para posterior visualização na interface.

**API**

A API foi desenvolvida em PHP e exposta localmente no Raspberry Pi. A estrutura de endpoints inclui:

GET /api/api,php?nome ={sensor}&valor – retorna o último valor do sensor;

GET /api/logs.php?sensor ={sensor} – devolve o histórico completo;

POST /api/api,php?nome ={sensor}&nometxt – permite receber o nome .

A API foi essencial para permitir que a interface web apresentasse os dados de forma dinâmica e separasse a lógica de apresentação da lógica de recolha e controlo.

**Interface Web**

A interface foi construída com HTML, CSS e PHP, recorrendo ao Bootstrap para garantir responsividade em dispositivos móveis. Esta inclui:

Dashboard com cartões para cada sensor/atuador (nome, valor atual, estado);

Histórico de valores com ordenação cronológica e gráficos básicos (por exemplo com Chart.js);

Autenticação de utilizador (página de login, controlo de sessões via PHP);

Página de administração com opções para ativar/desativar manualmente os atuadores.

**Organização do Projeto**

Cada sensor tem uma pasta própria no diretório api/, organizada da seguinte forma:

```
api/
│
├── temperatura/
│   ├── valor.txt
│   ├── hora.txt
│   ├── nome.txt
│   └── log.txt
├── humidade/
│   ...
├── ultrassonico/
│   ...
```

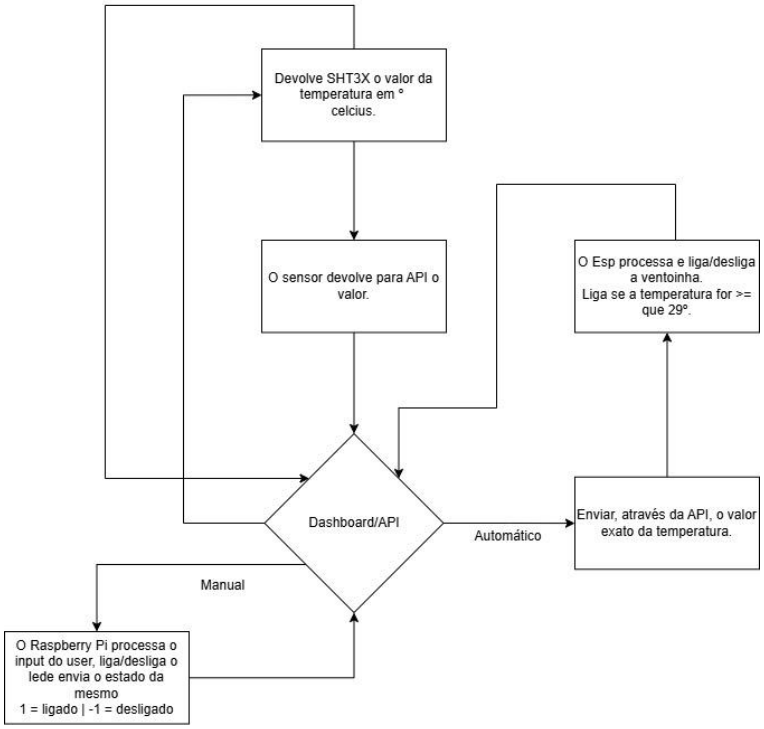


**Figura 2 - Diagrama para representação do sensor de temperatura**

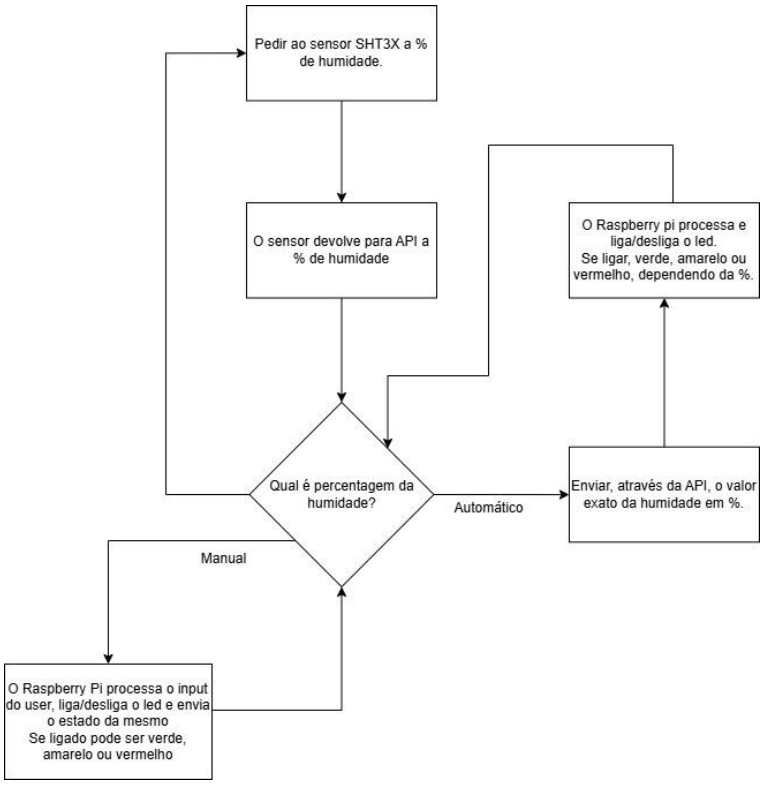


**Figura 3 - Diagrama para representação do sensor de humidade**

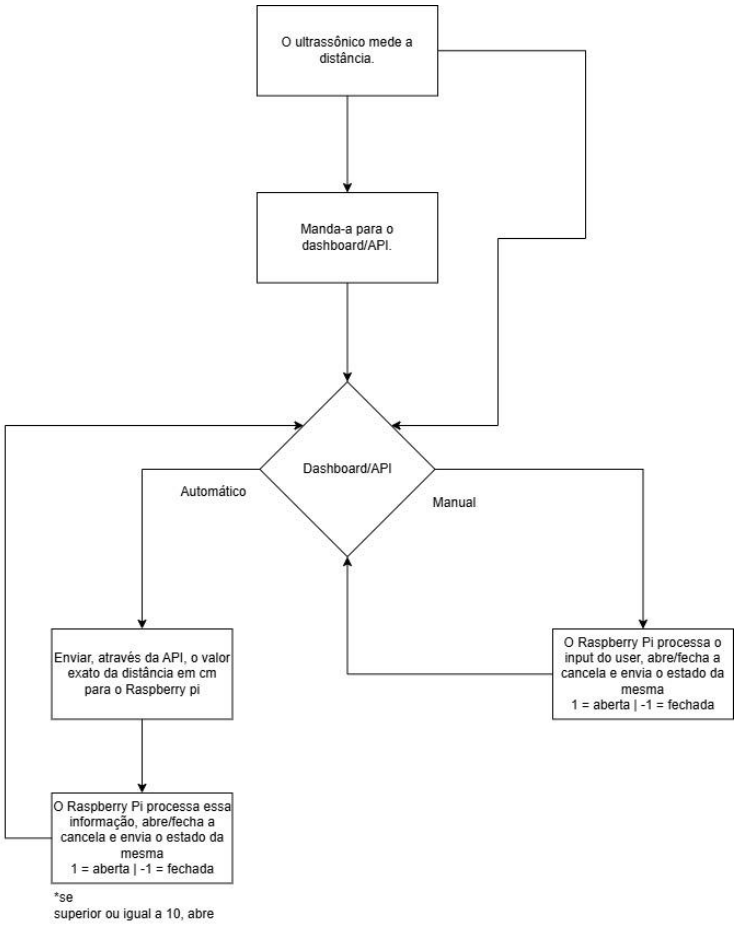


**Figura 4 - Diagrama para representação do sensor de ultrassônico**

# 4. Cenário de Teste

O cenário de teste corresponde a uma implementação funcional em pequena escala da arquitetura do supermercado inteligente, com todos os componentes interligados localmente através de um Raspberry Pi 5 e de um Esp32. O objetivo foi validar o funcionamento dos sensores, atuadores, API e interface web, garantindo que os eventos definidos são corretamente detetados e processados.

**Hardware Utilizado**

- Raspberry Pi 5 (sistema central de recolha e controlo)
  - Sensor ultrassónico HC-SR04 (GPIO 20 e 21)
  - Servo motor SG90 (GPIO 12)
  - LED RGB: vermelho, verde e amarelo (GPIOs 5 e 6 e 13, respetivamente)
- Esp32 (sistema central de recolha e controlo)
  - Sensor de temperatura e humidade SH3X (Pin 32 e 33)
  - Ventoinha (LED) (Pin 13,)

As portas GPIO mencionadas podem ser ajustadas consoante o hardware disponível, mas são as utilizadas no protótipo.

**Software Utilizado**

- Sistema operativo: Raspberry Pi OS (64 bits)
- Linguagem de programação: Python 3 (para scripts dos sensores/atuadores)
- Servidor web local: Apache2 com PHP
- Front-end: HTML, CSS, Bootstrap
- Back-end: PHP
- Outros: Visual Studio Code, para programação do site e dos scripts em C e Python. Usadas também bibliotecas como RPi.GPIO, request, csv, datetime e time

**Procedimentos Realizados**

**Configuração dos sensores:**

- Leitura periódica dos dados a cada 5 segundos.
- Escrita em ficheiros locais (valor.txt, hora.txt, log.txt) organizados por pasta.

**Configuração dos atuadores:**

- Scripts Python e C reagem automaticamente a eventos (por exemplo, temperatura >= 29 °C, liga a ventoinha).
- Os atuadores também podem ser acionados manualmente via interface web.

**Desenvolvimento da API local em PHP:**

- Permite aceder ao estado dos sensores e controlar atuadores.
- Utilizada pela interface web para apresentar dados em tempo real.

**Implementação da interface web:**

- Testada em servidor providenciado pela escola.
- Inclui login, dashboard, gráficos e histórico de leituras.

**Testes de funcionalidade:**

- Foram simuladas situações com diferentes temperaturas e distâncias no sensor ultrassónico.
- Confirmou-se o registo de eventos e a resposta dos atuadores.
- Testou-se a câmara e confirmou-se o funcionamento da mesma.

# 5. Resultados obtidos

Para validar o funcionamento da solução implementada, foram realizados vários testes no cenário de teste, com o objetivo de verificar o cumprimento dos eventos definidos, a fiabilidade da recolha de dados, o controlo dos atuadores e a integração entre os componentes (scripts Python/C, API, interface web).

**Teste 1 – Reação à temperatura elevada**

Objetivo: Verificar se a ventoinha é ativada automaticamente quando a temperatura ultrapassa os 29 °C.

Procedimento: Simulação de um ambiente quente próximo do sensor de temperatura.

Resultado: A ventoinha foi acionada durante 3 segundos. O evento foi registado corretamente no log.txt.

Estado: Sucesso

**Teste 2 – Deteção de presença via ultrassónico**

Objetivo: Confirmar que o servo motor reage à deteção de presença (distância inferior/igual a 10 cm).

Procedimento: Aproximação de um objeto ao sensor ultrassónico.

Resultado: O servo motor “abriu a porta”. Evento registado no log.txt.

Estado: Sucesso

**Teste 3 – Atualização dos ficheiros .txt**

Objetivo: Garantir que os ficheiros valor.txt, hora.txt e log.txt de cada sensor são atualizados corretamente.

Procedimento: Observação dos ficheiros após variação dos sensores.

Resultado: Todos os ficheiros foram atualizados com os valores e timestamps corretos.

Estado: Sucesso

**Teste 4 – Funcionamento da API**

Objetivo: Verificar se os endpoints da API retornam os dados corretos.

Procedimento: Testes com GET e POST via browser e ferramenta de testes (Postman).

Resultado: A API respondeu corretamente a todos os pedidos, incluindo a leitura de valores e ativação manual dos atuadores.

Estado: Sucesso

**Teste 5 – Interface Web**

Objetivo: Validar a exibição de dados em tempo real e a visualização do histórico.

Procedimento: Acesso à interface web com simulação de acesso, com diferentes utilizadores, eventos e alteração dos ficheiros.

Resultado: Os valores foram apresentados corretamente, com atualização dinâmica e visualização em formato de lista e gráfico.

Estado: Sucesso

**Teste 6 – Reação à temperatura elevada**

Objetivo: Verificar se a o Led RGB muda de cor (verde, amarelo e vermelho) a variância da humidade.

Procedimento: Simulação de um ambiente húmido e seco próximo do sensor de humidade.

Resultado: O Led mudou de cor várias vezes. O evento foi registado corretamente no log.txt.

Estado: Sucesso

**Considerações Finais**

Todos os testes foram bem-sucedidos e demonstraram a eficácia da solução desenvolvida. A comunicação entre sensores, API e interface web mostrou-se estável. A resposta dos atuadores foi rápida e precisa, e a estrutura de registo em ficheiros simples revelou-se funcional para o propósito do projeto.

# 6. Conclusão

O projeto desenvolvido permitiu criar uma solução IoT funcional e ajustada ao conceito de um supermercado inteligente, utilizando tecnologias simples, acessíveis e de fácil manutenção. A implementação de sensores, atuadores, uma API local e uma interface web responsiva permitiu monitorizar e controlar o ambiente de forma eficiente, atingindo todos os objetivos propostos.

Ao longo do desenvolvimento, foram enfrentados e ultrapassados desafios relacionados com a comunicação entre o hardware e o software, bem como na organização e acesso aos dados via ficheiros .txt. Ainda assim, a estrutura final revelou-se estável e eficaz, permitindo o registo contínuo de dados e a execução de ações automáticas com base em eventos predefinidos.

**A solução demonstrou o seu potencial em vários cenários de teste:**

- As leituras dos sensores foram fiáveis e consistentes;
- Os atuadores responderam corretamente aos estímulos definidos;
- A interface web mostrou-se funcional, intuitiva e informativa;
- A API local garantiu a separação lógica entre os dados e a sua apresentação.

**Pontos fortes:**

- Simplicidade da arquitetura e facilidade de manutenção;
- Modularidade do código e organização por pastas;
- Interface web clara e compatível com dispositivos móveis;
- Cumprimento de todos os requisitos da entrega.

**Possíveis melhorias futuras:**

- Substituição do sistema de ficheiros por uma base de dados (ex: SQLite ou MySQL);
- Integração com serviços cloud para acesso remoto ao painel;
- Adição de notificações (email/SMS) em caso de eventos críticos;
- Mais sensores e zonas para cobertura de todo o supermercado.

Em suma, o projeto cumpriu os objetivos definidos na introdução e proporcionou uma experiência prática valiosa na aplicação de conceitos de Internet das Coisas, integração de hardware com software e desenvolvimento web.

# 7. Bibliografia

Chandrasekar, P. (s.d.). *Stack overflow*. Obtido de Stack overflow: www.stackoverflow.com/

Microsoft. (s.d.). *Visual Studio Code*. Obtido de Visual Studio Code: www.code.visualstudio.com/

Thomas Dohmke. (s.d.). *Github*. Obtido de Github: www.github.com