



## **Estufa Inteligente**

Gabriel Ferreira Delgado Bernardes

David Roxo Fialho

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, junho de 2024

# Lista de Figuras

Figura 1 - Arquitetura IoT adaptada ao projeto .....	8
Figura 2 - Index.php .....	9
Figura 3 - Histórico.php .....	9
Figura 4 – Dashboard.php .....	10
Figura 5 – Clima.php.....	11
Figura 6 – grafico.php .....	12
Figura 7 – historicoimages.php .....	12
Figura 8 – Demonstração do efeito da página accounts.php .....	13
Figura 9 – Diagrama do atuador ar condicionado .....	14
Figura 10 – Cenário de Teste .....	16

## Lista de siglas e acrónimos

API	Application Programming Interface
CSS	Cascading Style Sheets
ESTG	Escola Superior de Tecnologia e Gestão
HTML	HyperText Markup Language
IoT	Internet of Things
IPLeiria	Instituto Politécnico de Leirias
MCU	Microcontrolador
PHP	PHP Hypertext Preprocessor
SBC	Single Board Computer

# Índice

Trata-se de um elemento **obrigatório**. Nota: **o índice nunca figura do índice**.

<b>Lista de Figuras .....</b>	<b>2</b>
<b>Lista de siglas e acrónimos.....</b>	<b>3</b>
<b>1. Introdução.....</b>	<b>4</b>
<b>2. Arquitetura .....</b>	<b>6</b>
<b>3. Implementação .....</b>	<b>14</b>
<b>4. Cenário de Teste .....</b>	<b>16</b>
<b>5. Resultados obtidos.....</b>	<b>17</b>
<b>6. Conclusão .....</b>	<b>18</b>
<b>7. Bibliografia .....</b>	<b>19</b>

# 1. Introdução

O tema “estufa inteligente” reflete a crescente importância da tecnologia na otimização de práticas agrícolas. Este desempenha um papel fulcral na vida humana.

As Estufas inteligentes abrangem quer o cultivo de plantas ornamentais quer a produção de floricultura, entre outros, sendo inúmeros os benefícios que podem ser adquiridos. Dada a elevada relevância na vida humana, decidimos abordar este tema e, naturalmente, explorá-lo.

Com o foco nesse objetivo, criámos uma forma de simular o ambiente de uma estufa e avaliar as condições nela existentes, como a luz, temperatura e humidade. As Estufas Inteligentes utilizam sensores e atuadores configurados para controlar as condições necessárias para o desenvolvimento das plantas lá existentes. Este controlo mais rigoroso, pode ter como resultado o aumento das colheitas e da qualidade, assim como a redução do desperdício de recursos naturais.

O trabalho encontra-se estruturado em duas partes, correspondentes às entregas. Na primeira parte, foi desenvolvido o HTML, CSS e PHP, e sobretudo a API. Já na segunda entrega, foi desenvolvida a comunicação entre os diferentes dispositivos do trabalho: arduino, raspberry, sem esquecer as funcionalidades extras.

## 2. Arquitetura

A arquitetura do nosso projeto envolve a utilização de diversos sensores e atuadores conectados a um MCU e a um SBC, que comunicam entre si e com a API central. A API serve como um intermediário para armazenar dados dos sensores e controlar os atuadores, além de fornecer informações para o servidor.

### 2.1 Desenvolvimento Inicial: HTML, CSS e PHP

O primeiro passo no desenvolvimento do nosso projeto foi a criação do próprio site, utilizando HTML, CSS e PHP. O objetivo já alcançado seria colocar valores no site de acordo com a informação adquirida nos sensores e atuadores, permitindo aos utilizadores visualizar em tempo real todos os dados. O HTML foi utilizado para estruturar o conteúdo da página, o CSS para estilização e o PHP para a lógica do servidor e comunicação com a API, assim como desenvolver funcionalidades extras, como cargos de acesso.

### 2.2 API

Com a dashboard concluída, o próximo passo foi desenvolver a API. A API foi desenhada com o intuito de ser uma intermediária eficiente entre os dispositivos (MCU e SBC) e o servidor. A API foi configurada para receber dados dos sensores por POST, armazenar essas informações na diretoria API/files do projeto, e disponibilizar os dados para a dashboard e histórico, e para os atuadores por GET. Assim, na página principal, é constantemente feito o GET da informação contida nos ficheiros, sendo posteriormente apresentada na página. De referir que, consoante o valor obtido na API, existem imagens diferentes.

### 2.3 MCU e SBC

Após a criação da API, focámos na realização da componente física do projeto, nomeadamente o Arduino (MCU) e a Raspberry Pi (SBC).

O MCU foi configurado para ler os dados dos sensores, como temperatura e humidade, através do sensor DTH, e enviar essas informações para a API pelo método POST. Além disso, fez o GET dos valores da API para controlar os atuadores, por exemplo, a ligar e desligar LED's.

A SBC foi configurada para fazer requisições GET à API para obter os dados dos sensores e, consequentemente, ter alguma ação nos atuadores, e com o POST para enviar os valores do sensor conectado.

A interação entre os dois dispositivos será detalhada no tópico 3.

## 2.4 Câmara

Por fim, o projeto foi complementado com a adição de uma câmara e outras funcionalidades extras. A câmara foi ligada à Raspberry Pi, permitindo a captura e transmissão de imagens em tempo real para a dashboard. Adicionalmente, também foi criado um histórico de imagens, e outras funcionalidades, detalhadas no subtópico seguinte.

## 2.5 Funcionalidades extras

As funcionalidades extras foram concebidas para que fosse possível melhorar mais o nosso projeto. Para tal, foi desenvolvido um histórico das imagens capturadas pela câmara, e o dark-mode. Além disso, foram usados dois atuadores adicionais e foi implementado o controlo das imagens, para que apenas sejam válidas as imagens com tamanho inferior ou igual a 1000kb, com formato jpg ou png. Para complementar, foram criados três cargos com privilégios diferentes. Também foi criada uma página extra que mostra o clima em Leiria ao longo do dia e uma página que mostra um gráfico com os valores dos sensores ao longo do tempo, com o objetivo de adequar as condições internas às necessidades das plantas.

## 2.6 Arquitetura

De seguida está representada a figura da arquitetura completa do projeto, destacando a interação entre os diferentes componentes e a API.

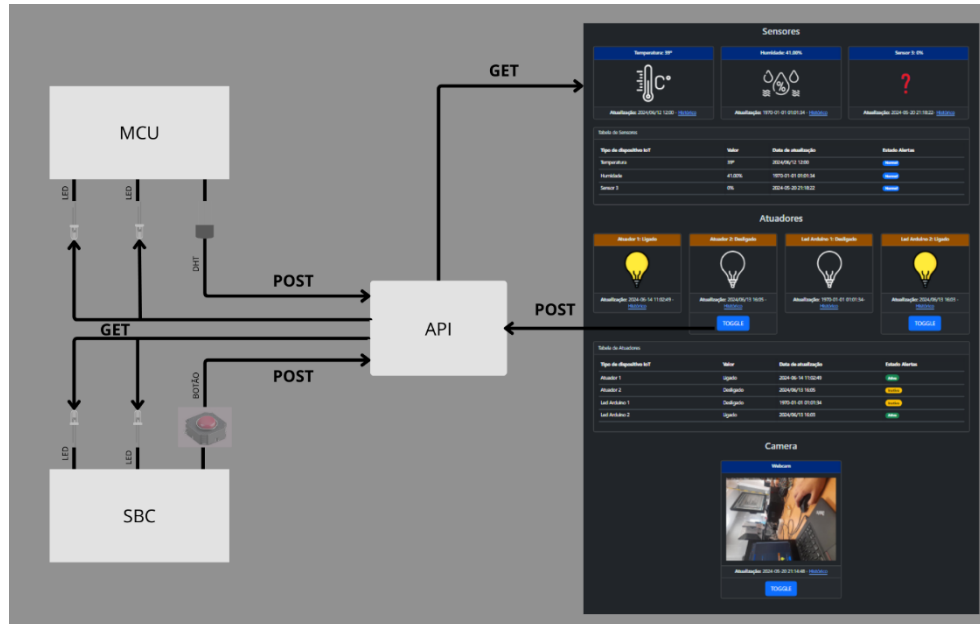


Figura 1 - Arquitetura IoT adaptada ao projeto



## 2.7 Descrição dos sites

### Index.php

Nesta página, é necessário realizar o login com as credenciais presentes no ficheiro ‘CONTAS.txt’, onde apenas se forem inseridos o username e password corretos, será permitido o acesso às outras páginas. Caso o utilizador tente entrar no site, mudando o URL para a dashboard ou qualquer outra página, não conseguirá se não efetuou login.

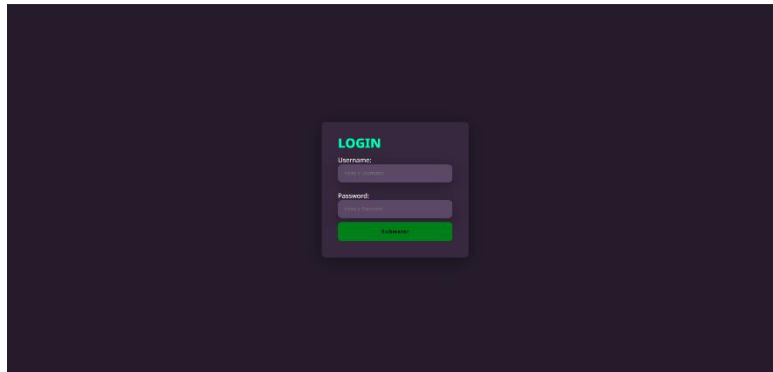
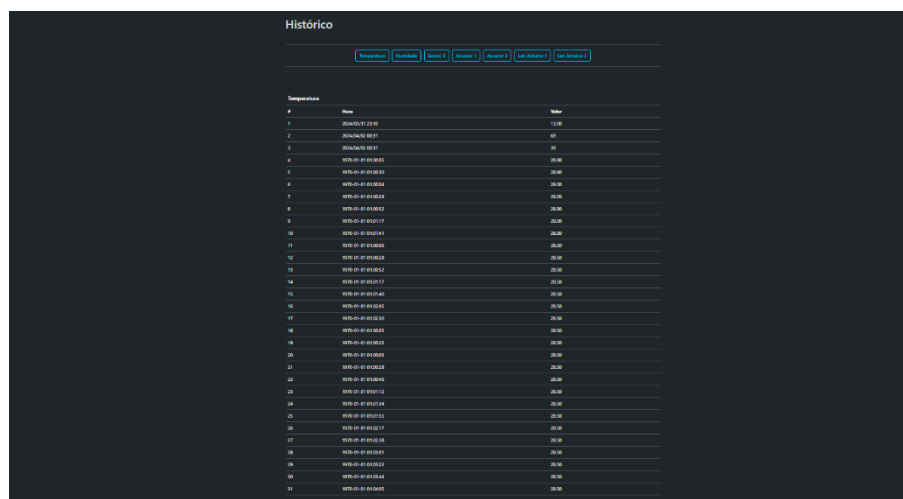


Figura 2 - Index.php

### Historico.php

Nesta página, cada sensor e atuador da estufa tem o seu próprio histórico. Inicialmente, todos os valores armazenados são apresentados, retirados dos arquivos log.txt respetivos dos sensores ou atuadores. É possível visualizar o histórico detalhado de qualquer sensor ou atuador específico.



ID	Data	Valor
1	2024-01-01 12:00:00	15.00
2	2024-01-01 12:05:00	15.00
3	2024-01-01 12:10:00	15.00
4	2024-01-01 12:15:00	15.00
5	2024-01-01 12:20:00	15.00
6	2024-01-01 12:25:00	15.00
7	2024-01-01 12:30:00	15.00
8	2024-01-01 12:35:00	15.00
9	2024-01-01 12:40:00	15.00
10	2024-01-01 12:45:00	15.00
11	2024-01-01 12:50:00	15.00
12	2024-01-01 12:55:00	15.00
13	2024-01-01 13:00:00	15.00
14	2024-01-01 13:05:00	15.00
15	2024-01-01 13:10:00	15.00
16	2024-01-01 13:15:00	15.00
17	2024-01-01 13:20:00	15.00
18	2024-01-01 13:25:00	15.00
19	2024-01-01 13:30:00	15.00
20	2024-01-01 13:35:00	15.00
21	2024-01-01 13:40:00	15.00
22	2024-01-01 13:45:00	15.00
23	2024-01-01 13:50:00	15.00
24	2024-01-01 13:55:00	15.00
25	2024-01-01 14:00:00	15.00
26	2024-01-01 14:05:00	15.00
27	2024-01-01 14:10:00	15.00
28	2024-01-01 14:15:00	15.00
29	2024-01-01 14:20:00	15.00
30	2024-01-01 14:25:00	15.00

Figura 3 - Histórico.php

## Dashboard.php

A dashboard atua como um ponto de comunicação entre o site e a API, que permite a monitorização em tempo real, controlando as condições da estufa de forma eficiente.

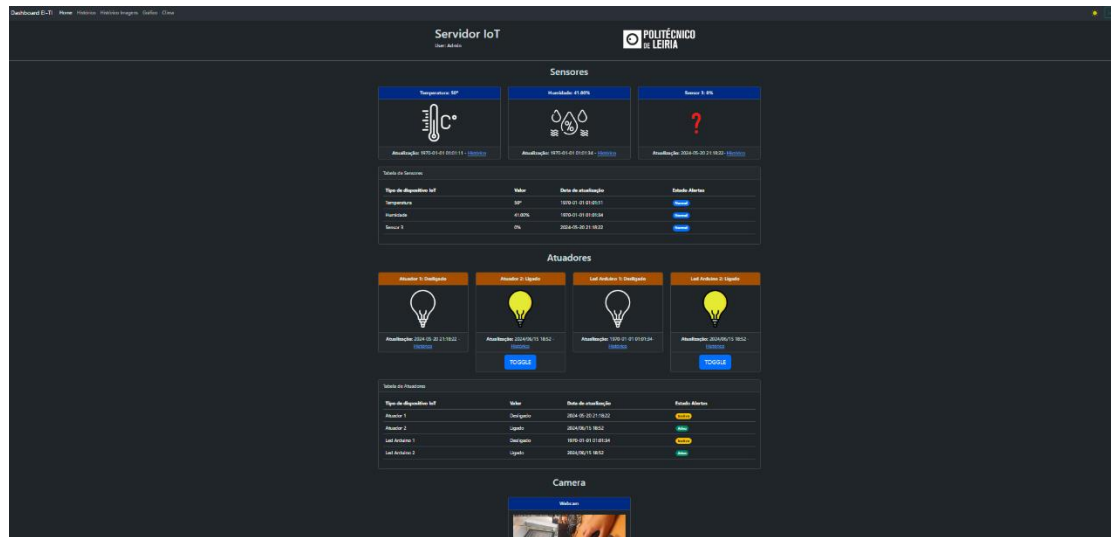


Figura 4 – Dashboard.php

## Clima.php

Integra a API da WeatherAPI, esta página foi idealizada com o objetivo de conhecer as condições metereológicas fora da estufa, com vista a adequar o clima existente na estufa. A API, embora paga, oferece um período de teste gratuito de 30 dias. Aqui, os dados são organizados numa tabela que detalha as condições. Este recurso é essencial para prever e mitigar os impactos climáticos no interior da estufa, prevenindo possíveis perigos.

Hora	Temperatura (°C)	Descrição	Humidade (%)
00:00	12.8	Partly Cloudy	91
01:00	12.6	Partly Cloudy	91
02:00	12.4	Partly Cloudy	91
03:00	12.4	Partly Cloudy	91
04:00	11.9	Partly Cloudy	91
05:00	11.7	Partly Cloudy	91
06:00	12.3	Partly Cloudy	87
07:00	14.3	Partly Cloudy	76
08:00	16	Partly Cloudy	67
09:00	17.5	Sunny	60
10:00	18.5	Sunny	55
11:00	19	Sunny	54
12:00	19.4	Sunny	55
13:00	19.6	Sunny	56
14:00	19.4	Sunny	58
15:00	19.1	Sunny	60
16:00	18.4	Sunny	62
17:00	17.7	Sunny	65
18:00	16.7	Sunny	69
19:00	15.7	Partly Cloudy	76
20:00	14	Partly Cloudy	84
21:00	13.3	Partly Cloudy	88
22:00	13	Partly Cloudy	90
23:00	12.8	Partly Cloudy	92

Figura 5 – Clima.php

## Grafico.php

A visualização de dados se torna uma experiência mais fácil com o uso da biblioteca Chart.js. Esta página permite ao usuário selecionar, via GET, qual sensor ou atuador deseja visualizar em formato gráfico. Os dados são extraídos dos log's específicos e apresentados de maneira clara e gráfica, facilitando a análise de tendências e comportamentos.

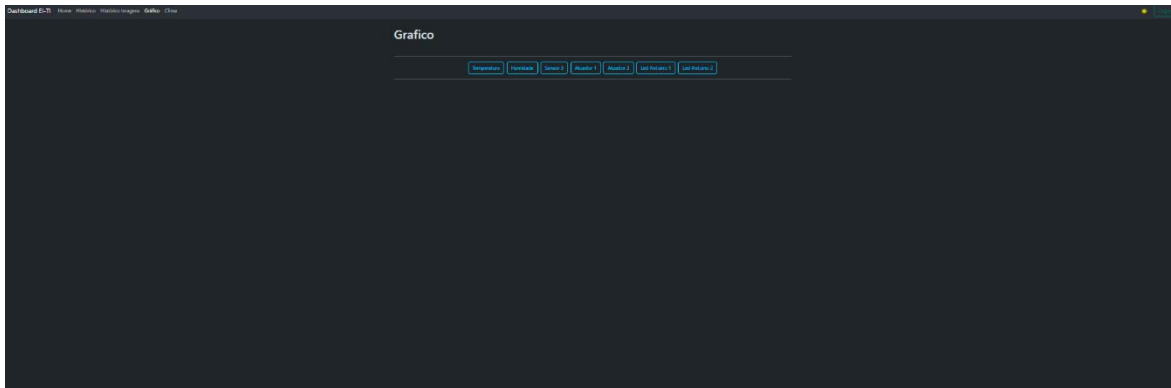


Figura 6 – grafico.php

## Historicoimages.php

Nesta páginas, a diretoria /historicoImages é explorada para exibir todas as fotos armazenadas, exceto a mais recente. Este recurso visual é uma forma prática de acompanhar o desenvolvimento da estufa, ao registrar cada momento significativo.

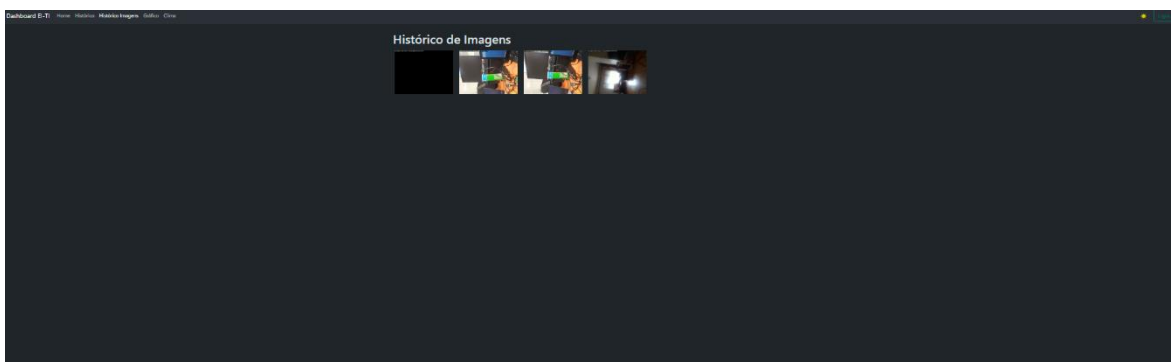


Figura 7 – historicoimages.php

## Upload.php

Com um foco na segurança e na organização, esta página de upload permite apenas imagens nos formatos PNG ou JPG, com um tamanho máximo de 1000KB.

## Account.php

A gestão de permissões é tratada com grande cuidado, oferecendo três níveis de acesso: user, root e viewer. O ROOT possui controlo total sobre todas as funcionalidades. O USER tem acesso e pode interagir com a dashboard, enquanto o VIEWER tem acesso restrito, podendo apenas visualizar a dashboard. Repare-se na figura a seguir, com a utilização de uma conta viewer, onde se pode ver que este cargo não tem permissões para ligar os atuadores, em relação à figura 3, onde a conta tem cargo admin.

The screenshot displays the 'Servidor IoT' dashboard for a user named 'viewer'. The dashboard is divided into two main sections: 'Sensores' (Sensors) and 'Atuadores' (Actuators).

**Sensores Section:**

- Three sensor cards are shown: 'Temperatura: 50°', 'Humidade: 41.00%', and 'Sensor 3: 0%'. Each card includes an icon, the current value, and a 'Histórico' link.
- Below the cards is a table titled 'Tabela de Sensores' with the following data:

Tipo de dispositivo IoT	Valor	Data de atualização	Estado Alertas
Temperatura	50°	1970-01-01 01:01:11	<span>Normal</span>
Humidade	41.00%	1970-01-01 01:01:34	<span>Normal</span>
Sensor 3	0%	2024-05-20 21:18:22	<span>Normal</span>

**Atuadores Section:**

- Four actuator cards are shown: 'Atuador 1: Desligado', 'Atuador 2: Ligado', 'Led Arduino 1: Desligado', and 'Led Arduino 2: Ligado'. Each card includes an icon, the current status, and a 'Histórico' link.
- Below the cards is a table titled 'Tabela de Atuadores' with the following data:

Tipo de dispositivo IoT	Valor	Data de atualização	Estado Alertas
Atuador 1	Desligado	2024-05-20 21:18:22	<span>Normal</span>
Atuador 2	Ligado	2024/06/15 18:52	<span>Ativo</span>
Led Arduino 1	Desligado	1970-01-01 01:01:34	<span>Normal</span>
Led Arduino 2	Ligado	2024/06/15 18:52	<span>Ativo</span>

Figura 8 – Demonstração do efeito da página accounts.php

### 3. Implementação

O desenvolvimento do projeto teve também como objetivos os eventos, ou seja, garantir uma boa comunicação entre todos os componentes usados. Em seguida serão detalhados.

Primeiramente, o sensor de temperatura e humidade presente no MCU faz o POST dos valores para a API. A SBC faz o GET do valor da temperatura e, se este não estiver compreendido entre os 30 e 60 graus, irá ser despoletada uma ação em Python para que o led ligue, simulando o ar condicionado a ser acionado devido às temperaturas inadequadas para o crescimento das plantas (Figura 2).

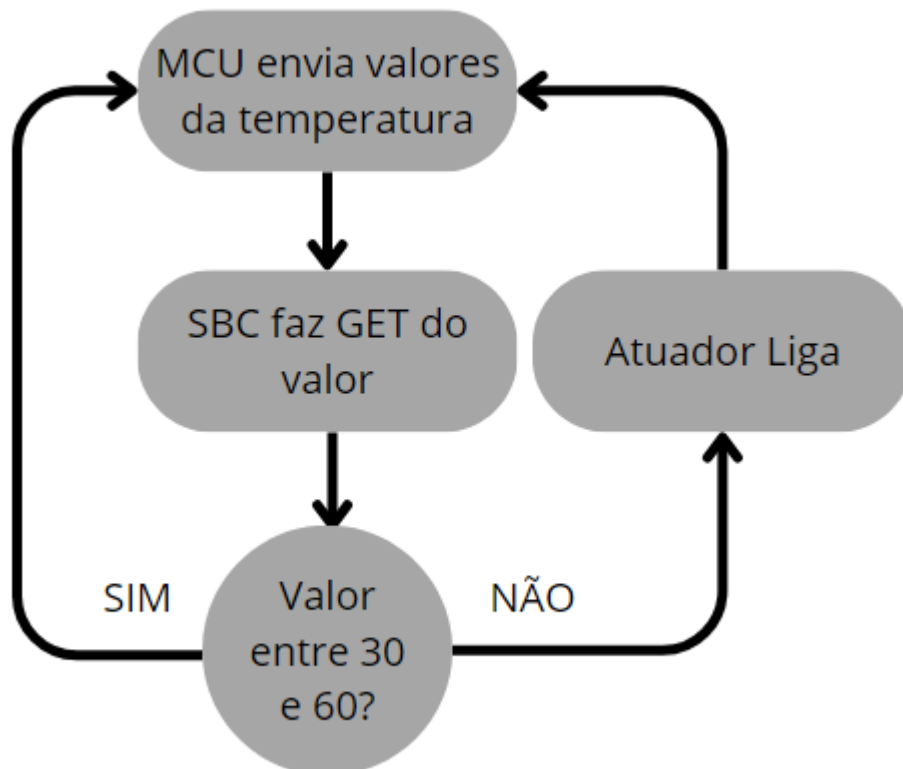


Figura 9 – Diagrama do atuador ar condicionado

Seguidamente, esses valores recolhidos pelo sensor da temperatura (e humidade) são atualizados através de GET's da Dashboard para obter os valores contidos nos ficheiros.

Posteriormente, o SBC faz o POST dos valores do sensor botão e, na MCU, é realizado o GET desses valores, que podem ou não ter como consequência uma ação no código C. São igualmente apresentados os valores do sensor na Dashboard.

Por último, foram adicionados três botões que proporcionam funcionalidades adicionais ao sistema.

O primeiro botão é o da câmera, que ao ser acionado, redireciona para 'usarCamera.php'. Este script altera o valor do parâmetro GET na API 'upload.php', ativa a câmera e captura uma nova foto. A foto antiga é renomeada e movida para a pasta do histórico de imagens, onde fica armazenada e passa a aparecer na página 'históricoimages.php'.

Os outros dois botões foram implementados usando JavaScript, com a utilização de um EventListener. Quando um destes botões é pressionado, o EventListener deteta o evento e envia uma requisição POST para a API, alterando o estado do atuador correspondente. O valor do estado muda de "desligado" para "ligado" ou de "ligado" para "desligado", conforme o estado atual. Esta abordagem assegura que as ações sejam realizadas de forma eficiente e em tempo real, proporcionando um controlo imediato sobre os atuadores conectados ao sistema.

## 4. Cenário de Teste

As entregas foram sempre antecedidas por testes, com o objetivo de verificar o bom funcionamento de todos os requisitos e funcionalidades. Para tal, foram usados, de forma a verificar tudo para a primeira entrega, o RESTer para verificar o bom funcionamento da API, e a simulação de vários dispositivos para verificar a responsividade. Na segunda entrega, recorremos ao MKR1000 (MCU) e ao Raspberry Pi (SBC), assim como à utilização do sensor DHT11, do botão, e dos 4 led's como atuadores, a simularem um alarme, um ar condicionado, a rega e um servomotor para abrir e fechar a porta.

Foi programado código em Python na SBC e código em C na MCU de forma a controlar os atuadores através de pedidos GET. Também foram realizados pedidos POST de forma a enviar para a API os valores captados pelos sensores conectados aos dois dispositivos. Para configurar o SBC, primeiro foi necessário formatar um cartão SD e instalar o ambiente gráfico em Linux, de forma a ter acesso à linha de comandos da Raspberry e a ter a possibilidade de usar bibliotecas já instaladas na SBC, como por exemplo 'RPi.GPIO'.

Para a Câmara foi utilizada a aplicação DROIDCAM que permite utilizar a câmara do telemóvel, criando uma IP cam que é usada para o SBC ir buscar a foto. Esta foto apenas é aceite se for inferior ou igual a 1000 kb no formato jpg ou png.

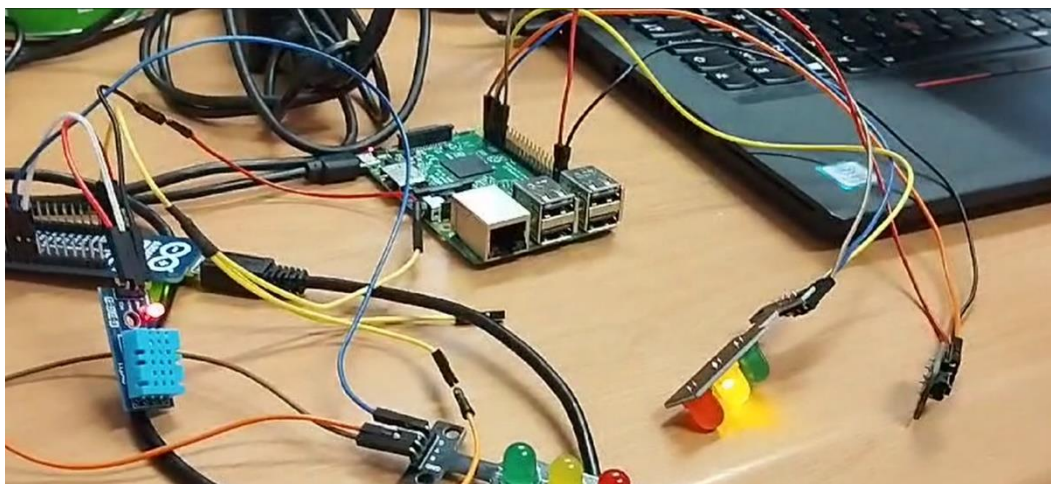


Figura 100 – Cenário de Teste



## 5. Resultados obtidos

Agora serão apresentados os resultados dos testes realizados para comprovar que os objetivos foram alcançados.

Primeiramente, verificamos o bom funcionamento da API, que é fundamental para a comunicação entre os diferentes componentes. Utilizámos a extensão RESTer para enviar requisições GET e POST, e confirmámos que a API se comportava conforme esperado. Em seguida, o Dashboard, como esperado, faz o GET desses valores e apresenta-os ao utilizador.

Para assegurar uma boa comunicação entre o MCU, o SBC, e a API, configurámos o MCU para ler dados do sensor de temperatura e humidade (DHT11) e enviá-los para a API. O SBC foi configurado para obter esses dados da API e ligar os LED's conforme necessário. Realizámos simulações de diferentes condições de temperatura e humidade para observar o comportamento do sistema. Da mesma forma, o SBC lê os dados do botão e envia-os para a API. O MCU obtém esses dados e liga ou desliga o led. Assim, fica demonstrada a comunicação eficiente entre os dispositivos.

Além disso, testámos a responsividade do projeto. Usámos uma ferramenta da GOOGLE para testar a responsividade em diferentes dispositivos. O site correspondeu às expectativas.

Seguidamente, verificámos o bom funcionamento da câmara que, quando o script em Python está ativo, e pressionamos o botão da Dashboard (apenas visível para o Admin e user), captura uma foto e disponibiliza-a na Dashboard.

Todos os testes realizados confirmaram que os objetivos do projeto foram alcançados, com a API a funcionar corretamente, a comunicação entre os dispositivos eficiente, entre outros.

## 6. Conclusão

Neste projeto, a implementação de uma estufa inteligente teve como objetivo a eficiência na comunicação de várias tecnologias para otimizar as condições ambientais do cultivo de plantas. Ao longo do desenvolvimento, atingimos os objetivos estabelecidos e verificamos a importância de cada componente na arquitetura geral do sistema.

A utilização de dispositivos tais como MCU e SBC, permitiu monitorizar e ajustar as condições ideais para o desenvolvimento, aumentando a eficiência. A API foi, sem dúvida, um elemento fulcral no desenvolvimento deste projeto. O ponto que consideramos mais fraco que poderia ser melhorado, é o facto de as contas estarem contidas num ficheiro de texto, sendo pouco seguro numa solução de maior escala.

O projeto cumpriu os seus objetivos, oferecendo ainda vastas funcionalidades extras, de forma a melhorar o trabalho. Esta solução reflete a importância da tecnologia na agricultura moderna, sendo uma ferramenta muito útil na otimização e na sustentabilidade.

## **7. Bibliografia**

PowerPoints da componente prática e teórica de Tecnologias de Internet.