Smart GreenHouse

1st Cláudio Barbosa Martins
Engenharia de Telecomunicações e Informática
a101279@alunos.uminho.pt

2nd Diogo Filipe Monteiro de Sousa Gonçalves
Engenharia de Telecomunicações e Informática
a89386@alunos.uminho.pt

3rd Eduardo de Sousa Sobral
Engenharia de Telecomunicações e Informática
a94157@alunos.uminho.pt

4th João Pedro Catela Fernandes
Engenharia de Telecomunicações e Informática
a93080@alunos.uminho.pt

5th Nuno Duarte Mirra
Engenharia de Telecomunicações e Informática
a89406@alunos.uminho.pt

***Abstract*—Este artigo descreve o desenvolvimento de um projeto de simulação de uma *smart greenhouse* no âmbito da Unidade Curricular de Projeto Integrado de Telecomunicações do 3º ano da Licenciatura de Engenharia de Telecomunicações e Informática da Universidade do Minho. Foram investigados sensores adequados, novas linguagens de programação e estratégias de gestão de tarefas para identificar a solução mais eficaz. O sistema desenvolvido integra sensores para monitorização e uma interface web para gestão e controlo.**

***Index Terms*—smart greenhouse, IoT, Wi-Fi, ESP32**

I. Introdução

Nos dias de hoje com os avanços tecnológicos é cada vez mais fácil termos todas as comodidades em nossas casas. Uns dos principais objetivos das *smart greenhouses* é garantir conforto e praticidade aos seus utilizadores na hora de cuidar das suas plantas. O mesmo é possível devido à IoT (*Internet of Things*) que possibilita a ligação de vários dispositivos e sensores através da rede Wi-Fi, que por sua vez podem ser controlados a partir de diversos dispositivos, como *tablets*, *smartphones* ou mesmo um computador. [1]

Este artigo tem como objetivo apresentar o sistema completo criado para simularmos uma *smart greenhouse*. O sistema é composto por diversos elementos dos quais destacamos os sensores para monitorização e uma interface web desenvolvida para a gestão e controlo do sistema pelos utilizadores. Ao longo deste artigo, serão abordados detalhes sobre os componentes do sistema, as devidas funcionalidades e como os mesmos foram implementados para além disso serão discutidas algumas vantagens e desvantagens do nosso sistema.

Através deste projeto, evidenciamos a relevância no contexto dos dias que correm e o potencial de ter uma *smart greenhouse*.

II. Fundamentos/Trabalho Relacionado

As linguagens de progrmaçao utilizadas foram então as seguintes: o nosso sistema sensor foi implementado em C++ usando o ArduinoIDE para a execução, o gestor de serviço e o sistema sensor simulado foram implementados em Python, a base de dados relacional em SQL. Para o *web server* recorremos à framework Django, também escrita em Python, em conjunto com HTML e CSS para o *frontend*.

III. Projeto/Conceção do Sistema

Este projeto é constituído por 3 sistemas fundamentais, são eles os seguintes: Sistema Sensor, Sistema Central e a Interface *Web*.

Sendo este projeto um projeto que requer um leque vasto de tecnologias e sistemas diferentes, vamos dividir esta secção em várias subsecções, como a montagem do hardware e a sua programação, desenvolvimento de sistemas simulados, desenvolvimento de um gestor de serviços para comunicar com o sistema real, simulados e o respetivo protocolo de comunicação, base de dados SQL, construção de um *website* e uma API que trabalham em conjunto e servem de interface para os clientes. Para melhor perceção do sistema global na seguinte imagem encontra se a arquitetura do mesmo.

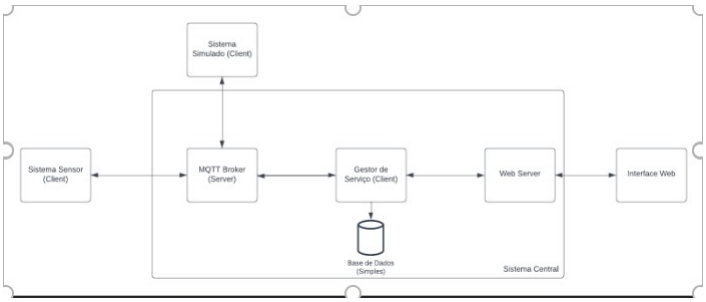

Fig. 1. Arquitetura do Projeto.

A. Sistema Sensor

Para a criação do sistema sensor criamos esta lista de aquisição do material necessário onde temos diversos sensores.

TABLE I
LISTA DO MATERIAL NECESSÁRIO

Quantidade	Nome
1	Placa ESP32-FireBeetle
1	Sensor DHT22
1	Sensor BH1750
1	Sensor Humidade Solo
1	Sensor Nivel de Líquidos
1	Breadboard
2	LEDs

1) DHT22: O DHT22 é um sensor capaz de ler valores de temperatura e de humidade no ar, simples e facil de utilizar com uma placa Arduino, tal como a ESP-32 FireBeetle. O sensor é capaz de medir temperaturas entre os -40 e 80 °C, tendo uma precisão de +- 0,5 °C. Em termos de humidade o sensor mede valores entre os 0 e 100% RH(*relative humidity*) tendo uma precisão de +-2%RH. Ambas as medições são realizadas em intervalos de 2 s.

2) BH1750: O BH1750 é um sensor de intensidade de luz. Este sensor usa comunicação I2C o que permite uma conexão fácil com a placa *FireBeetle*. Este sensor mede intensidades de luz entre 1 e 65535 lux com uma precisão de +- 1 lux. Como o sensor se encontra sempre em modo de alta resolução as medições têm intervalos de 120 ms.

3) Sensor humidade solo: Neste caso utilizamos um sensor de humidade do solo comum capaz de detetar a quantidade de água num pedaço de solo e apresentar esses dados em percentagem, sendo 0% de humidade solo completamente seco, e 100% solo ”encharcado”, com uma precisão de +-5%.

4) Sensor de Nível de Líquidos: Para a medição do nível da água foi utilizado um medidor de nível de líquidos, que por sua vez, também pode ser utilizado como sensor de chuva. Como a saída é analógica, a tensão de saída aumenta em função da profundidade de imersão do módulo, ou da quantidade de gotas de chuva que nele estão depositadas. Esses dados são apresentados em percentagem sendo 0% reservatório vazio e 100% reservatório cheio.

Todos os materiais e sensores previamente mencionados foram colocados numa *breadboard* e ligados à placa ESP-32 *FireBeetle*.O sensor DHT22 e o sensor BH1750 permanecem na *breadboard* para fazer as suas medições enquanto o sensor de humidade de solo e o sensor de nível da água são inseridos num vaso e num reservatório com água. Ao meter água no vaso os sensores efetuam as suas medições. Os LEDs são de duas cores diferentes, 1 verde e 1 vermelho. O LED verde acende quando o valor da humidade do solo é superior a 70% e o LED vermelho acende quando a humidade do solo é inferior a 70%.

B. Gestor de Serviço

De modo a criar o gestor de serviços necessitamos de uma ferramenta capaz de receber os dados que cada sistema

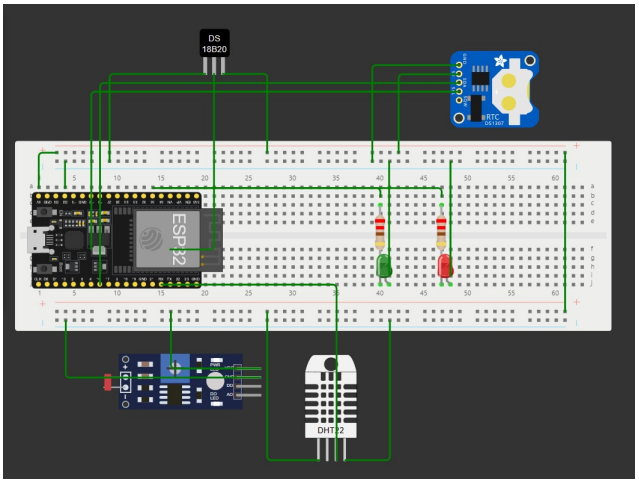

Fig. 2. Representação do Sistema Sensor.

comunica e distribui-los sem haver qualquer tipo de conflito de dados entre sistemas. Para tal implementamos um *broker* MQTT capaz de receber os dados dos diversos sistemas sensores e publicá-los num tópico que o Gestor de Serviços posteriormente subscreve e executa o tratamento de dados. O Gestor de Serviços é então capaz de compreender os dados recebidos do *broker* e envia-los para a base de dados.

C. Sistemas Simulados

Os sistemas simulados acima mencionados apenas serviram para conseguirmos recriar situações com vários utilizadores com diversos sistemas sensor. Para tal criamos um *script Python* capaz de recriar dados dentro de um certo intervalo para os diversos sensores de um sistema sensor e . Posteriormente esse sistema é capaz de ser atribuído a um utilizador registado.

D. Base de Dados

A base de dados recebe os dados recolhidos nos diversos sistemas que são guardados na tabela ’LEITURAS’. Foi também criada uma tabela de utilizadores, com o mesmo nome, que contém todos os utilizadores registados. Assim a base de dados guarda todas as medições das plantas associando esses dados aos respetivos utilizadores, recorrendo à tabela ”sistema_utilizador”. A base de dados responde também a pedidos de amostragem de dados em tempo real, pedidos esses realizados na página *web* acoplada ao sistema.

E. Página Web

Foi também criada uma página web de modo a ser possível ao utilizador monitorizar e analisar os dados das suas plantas. A página *web* tem um sistema de *login* capaz de diferenciar as credenciais de utilizadores e administradores de serviço, direcionando cada um para o seu respetivo menu inicial. É também capaz de apresentar os dados captados pelos sensores em tempo real e gerar gráficos dos mesmos, tanto como filtrar a amostragem desses mesmos dados.

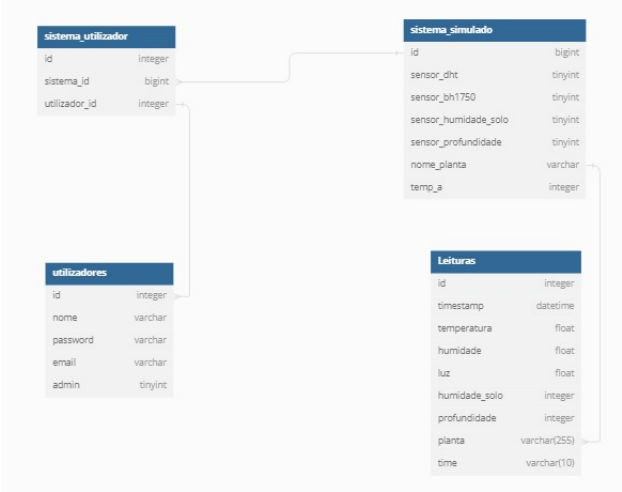

Fig. 3. Diagrama de entidade e relacionamento da base de dados.

Fig. 4. Pagina de *Login*.

F. Explicação da conexão dos blocos do sistema

O MQTT *broker* acaba por ser a peça fulcral neste projeto pois é pelo mesmo que passa toda a informação vinda de todos os sistemas envolvidos no projeto. Recebe os dados dos sensores e transmite-os para o Gestor de Serviços submetendo esses dados num tópico através do IP local. O Gestor de Serviços é então capaz de subscrever a esse tópico, receber os dados nele contidos. Analisando os dados, o Gestor de Serviços transmite os mesmos para a Base de Dados. O Gestor de Serviços é também responsável por interpretar os pedidos realizados pelo utilizador à base de dados na página *web* e fazer a transmissão dos mesmos de volta para a página.

IV. Implementação

A. Protocolo de Comunicação

Para estabelecermos a comunicação entre o Gestor de Serviços e o Sistema Sensor utilizou-se o protocolo MQTT,

Fig. 5. Pagina de Registo.

que permite o envio e a receção de dados entre ambos os sistemas. Este protocolo utiliza um *broker* que funciona como um intermediário entre o Sistema Sensor, que é o publicador, e o Gestor de Serviços que é o subscritor, que recebe os dados gerados.

V. Conclusão

Este projeto de simulação de uma *smart greenhouse* demonstrou a viabilidade e a eficiência de um sistema integrado de monitorização e controlo de plantas ou estufas. Utilizando sensores variados e tecnologias modernas, como a placa ESP32-FireBeetle e o protocolo MQTT, conseguimos criar uma solução robusta que permite a coleta e análise de dados em tempo real.

Os resultados obtidos evidenciam a capacidade do nosso sistema em monitorizar parâmetros críticos, como temperatura, humidade do ar e do solo, e níveis de luz e líquidos, oferecendo ao utilizador final uma ferramenta prática e eficaz para a gestão das suas plantas. A integração de uma interface web amigável, desenvolvida recorrendo à interface Django, HTML e CSS, facilita o acesso e o controlo dos dados, proporcionando uma experiência de utilizador aprimorada.

Além disso, a implementação de um *broker* MQTT para a comunicação entre os diferentes sistemas garantiu a fiabilidade e a eficiência na transmissão de dados, essencial para o funcionamento contínuo e em tempo real da *smart greenhouse*.

Em suma, o projeto alcançou seus objetivos ao proporcionar uma solução tecnológica inovadora e acessível para a monitorização automatizada de estufas, destacando-se como uma aplicação prática da Internet das Coisas (IoT) na agricultura. Futuras melhorias podem incluir a incorporação de uma aplicação móvel para melhor mobilidade na utilização e monitorização do sistema.

REFERENCES

[1] https://wiki.dfrobot.com/FireBeetle_ESP32_IOT_Microcontroller(V3.0)__Supports_Wi-Fi_&_Bluetooth__SKU__DFR0478
[2] https://www.djangoproject.com/
[3] https://www.emqx.com/en/blog/the-ultimate-guide-to-mqtt-broker-comparison
[4] https://docs.arduino.cc/software/ide/
[5] https://www.w3schools.com/html/
[6] https://www.w3schools.com/css/deafault.asp
[7] https://youtube.com