

## Sistema para monitoramento e automação de irrigação de hortas

### *System for monitoring and automation of vegetables garden irrigation*

Houssan Ali Hijazi 

Jefferson de Oliveira Chaves 

**Resumo:** A Internet das Coisas (IoT), pode ser descrita como uma rede de objetos físicos, que por meio de sensores, protocolos e outras tecnologias, conectam-se e trocam dados com outros dispositivos e sistemas pela internet. Dentre as áreas que mais se beneficiam, destacamos a agricultura. Atualmente, o uso sustentável dos recursos ambientais, como a água, é fator relevante para qualquer negócio e nesse sentido, esse trabalho se propõe a estudar, e implementar um sistema para monitoramento e automação do processo de irrigação em hortas. A automação de um sistema de irrigação, possibilitará que recursos como adubo, água e energia, sejam empregados de forma sustentável e racional. Espera-se que a prática de irrigação automatizada possa aumentar a produtividade, evitar o desperdício de água e diminuir custos na produção. A solução a ser implementada pretende auxiliar os cultivos de hortas domésticas, comunitárias, escolares e até mesmo pequenas propriedades rurais a diminuir o consumo de água por meio do monitoramento e automação do processo de irrigação. Isso será realizado a partir da obtenção de dados de sensores ligados a um microcontrolador, que serão persistidos em nuvem e monitorados via aplicativo de celular.

**Palavras-chave:** Internet das Coisas, ESP32, Irrigação, Automação, Monitoramento.

**Abstract:** The Internet of Things (IoT) can be described as a network of physical devices that, using sensors, protocols and other technologies, can connect and exchange data with other devices and systems over the Internet. Among the areas that have been benefited by such technology, agriculture is a highlight. Nowadays, the sustainable use of natural resources (like water) is a major factor in any business, and having that in mind, this work we present here studies and implements a system for automation and monitoring of small crops/ vegetable gardens irrigation. Such automation and monitoring will allow rational and sustainable use of resources like water, fertilizers and electricity. It's also expected an increase in overall productivity and decrease in loss or waste of resources, consequently, lowering production costs. The suggested solution focuses on small crops/vegetable gardens (homes, schools, community and small rural properties). The results of the proposed system and hypothesis will be achieved by collecting data from a diverse set of sensors connected to a microcontroller. All data will be persisted in a cloud environment and accessible via cellphone.

**Keywords:** Internet of things, ESP32, Irrigation, Automation, Monitoring.

# 1 INTRODUÇÃO

A agricultura é a atividade que objetiva a cultura do solo para produção de vegetais, visando, principalmente, assegurar a subsistência alimentar do ser humano, além de suportar a produção de matérias-primas que são transformadas em produtos em outros campos da atividade econômica (SEBRAE, 2017). Trata-se de uma das formas principais de transformação do espaço geográfico, sendo uma das mais antigas práticas realizadas na história. A agricultura se divide em diversos ramos, tais como pecuária, silvicultura e horticultura. O conceito moderno de horticultura a coloca como um ramo da agricultura, que trata do cultivo de diversos tipos de plantas, seja em hortas, estufas ou jardins (UEMG, 2018). De forma geral, a horticultura é o setor responsável pelo cultivo de plantas comestíveis e não comestíveis: frutas, hortaliças, verduras, legumes, flores e árvores.

A horticultura tem suas atividades realizadas em espaços chamados de hortas. Hortas são locais em que são cultivadas hortaliças e outras plantas, como ervas condimentares e aromáticas. Existem vários tipos de hortas, dependendo do tamanho, do número de hortaliças cultivadas e, principalmente, do objetivo, que varia da exploração comercial ao consumo doméstico. Alguns exemplos são as hortas comerciais, domésticas, institucionais, escolares e comunitárias, as do sistema de produção convencional ou orgânico e a horta autossustentável (JORGE *et al.*, 2016).

Ainda de acordo com Jorge *et al.*, (2016) hortas são locais em que são realizadas todas as atividades referentes à produção de hortaliças, ervas medicinais, temperos e outras plantas. As atividades envolvidas no cultivo de uma horta, incluem preparação do solo, adubação, plantio, manejo e colheita. A atividade de irrigação é uma das atividades essenciais para o cultivo e produção de uma horta.

Desde tempos pré-históricos, o homem vem praticando a irrigação desviando cursos de água para atender às necessidades de suas plantações (TESTEZLAF, 2017). De acordo com Barbosa (2013), a irrigação é o processo artificial de fornecer recursos hídricos para as plantas em quantidades

apropriadas, a fim de manter a umidade do solo necessária para o desenvolvimento das mesmas.

A irrigação empírica, sem controle da quantidade de água aplicada, pode acarretar vários problemas para as plantações. A irrigação com quantidades insuficientes de água pode reduzir o metabolismo das plantas, prejudicando o desenvolvimento e a produtividade, resultando em uma produção abaixo do esperado. Por outro lado, a aplicação de quantidades excessivas de água pode favorecer o surgimento de doenças e prejudicar as raízes das plantas, criando condições para a proliferação de fungos no solo e nas raízes, levando à decomposição e lavagem de nutrientes, além de ser um desperdício de um recurso natural (GUIMARÃES, 2011).

Diante deste cenário, produtores, buscando eficiência e sustentabilidade, passaram a procurar soluções para a automação de suas hortas (MAROUELLI; SILVA, 2011). Temporizadores, sistemas automatizados e outros conjuntos de ferramentas tecnológicas, como a Internet das Coisas, passaram a ser amplamente utilizados como suporte ao processo de cultivo, visando o uso racional dos recursos naturais, do uso do solo e de adubo, do uso de energia e principalmente o uso da água, por meio da automatização dos sistemas de irrigação.

Descrita como uma rede de objetos físicos, que por meio de sensores, protocolos e outras tecnologias, conectam-se e trocam dados com outros dispositivos e sistemas pela internet (AL-FUQAHA *et. al*, 2015), a Internet das Coisas (IoT) se destacou como tecnologia para automação e monitoramento de diversas atividades, apontam potencial para melhorias relacionadas ao aumento da produção, ao uso sustentável dos recursos naturais e da mão de obra na atividades de cultivo.

Diante do exposto, este trabalho se propõe a estudar, planejar e propor uma aplicação para monitoramento e automação do processo de irrigação em hortas. A automação de um sistema de irrigação, possibilitará que recursos sejam empregados de forma sustentável e racional. Espera-se que a prática de irrigação automatizada possa aumentar a produtividade, evitar o desperdício de água e diminuir custos na produção. A solução a ser implementada é uma

prova de conceito que pretende auxiliar os cultivos de hortas domésticas, comunitárias, escolares e até mesmo pequenas propriedades rurais a diminuir o consumo de água por meio do monitoramento e automação do processo de irrigação. Isso será realizado a partir da obtenção de dados de sensores ligados a um microcontrolador, que serão persistidos em nuvem e monitorados via aplicativo de celular.

Esse trabalho está organizado nas seguintes seções: materiais e métodos, desenvolvimento, resultados e considerações finais.

## 2 MATERIAIS E MÉTODOS

Ao se desenvolver uma pesquisa é necessário definir previamente quais serão as abordagens e os métodos adotados na condução do problema. Dessa forma, é preciso ter em mente alguns elementos no momento da escolha de sua abordagem epistemológica, bem como na escolha dos métodos da pesquisa. Dois elementos centrais que devem ser considerados nessa escolha são: a natureza do problema de pesquisa e os seus objetivos.

Ao considerar-se a natureza do problema deste trabalho e seus objetivos optou-se pela metodologia *Design Science Research* (DSR). A escolha por tal paradigma justifica-se, uma vez que seu propósito é buscar projetar e/ou produzir sistemas ou modificar os existentes (os chamados artefatos). Ainda, é orientada a prescrição, isto é, a solução de problemas (DRESCH, LACERDA, JÚNIOR, 2015).

Cabe ressaltar, que a DSR possui distintas propostas de operacionalização do método, entretanto, nesta pesquisa optou-se pela apresentada por Peffers *et al.* (2007). As etapas podem ser definidas da seguinte forma:

- Identificação do problema: justificar a importância da pesquisa, sua relevância e do problema investigado, e da aplicabilidade da solução proposta;

- Definição dos resultados esperados;
- Projeto e desenvolvimento do artefato: definição de suas funcionalidades, arquitetura e o desenvolvimento em si;
- Demonstração do artefato;
- Avaliação dos resultados;
- Comunicação do trabalho.

Seguindo o protocolo metodológico, após a identificação do problema e a definição dos resultados esperados, passou-se à etapa de Projeto e desenvolvimento do artefato. É importante destacar que o projeto em questão é composto por três aplicações que funcionam de forma orquestrada, trocando mensagens entre si, com o intuito de realizar o processo de monitoramento e irrigação automatizada: *i)* dispositivos físicos (*hardware*); *ii)* aplicação *backend* integrado com nuvem; e *iii)* uma aplicação para dispositivos móveis.

Assim, para alcançar os objetivos propostos, o projeto e implementação do sistema apresentado por este trabalho, consistiu, de forma resumida, na configuração e construção de um *hardware*, composto principalmente por um microcontrolador ESP32, sensores e atuadores. Foram adicionados sensores de temperatura e umidade do ar, sensores de chuva e umidade do solo instalados ao microcontrolador ESP32. O algoritmo utilizado pelo microcontrolador, foi programado com a linguagem MicroPython. Esse algoritmo utilizou-se do protocolo *Message Queuing Telemetry Transport* (MQTT), para enviar os dados obtidos para uma outra aplicação (*backend*), programada em linguagem Python, que persiste esses dados no *Firebase Firestore*. Além disso, essa aplicação ainda analisa tais dados e com base em seus valores, realiza operações automatizadas, tais como ligar ou desligar a irrigação. Esse processo é realizado por meio do envio de comandos da aplicação *backend* para o microcontrolador, também empregando o uso de MQTT. O monitoramento dos dados obtidos, bem como o controle do processo de irrigação pode ser realizado por meio de um aplicativo Android, desenvolvido como parte desse projeto.

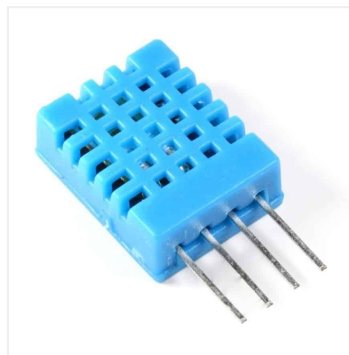
Detalhes da implementação do projeto serão descritas no tópico “Desenvolvimento”, deste trabalho. Os próximos tópicos apresentam as tecnologias, componentes de *hardware* e componentes de *software* utilizados neste projeto.

## 2.1 Sensor de temperatura e umidade do ar

O sensor de temperatura e umidade do ar (Figura 1) é um dos componentes mais utilizados em projetos que envolvem medição de temperatura e umidade do ar ambiente. Para este trabalho, optou-se pela utilização do sensor DHT11. Este sensor realiza medições de temperatura de 0° até 50° celsius e mede a umidade do ar nas faixas de 20% a 90%. A margem de erro desse sensor para medição de temperatura é de aproximadamente 2° celsius e para umidade é de 5%. O sensor de umidade é capacitivo e o sensor de temperatura é um termistor NTC, que é um resistor sensível à variações de temperatura (MURTA, 2019).

Os dados deste sensor foram utilizados para obter indicadores de umidade e temperatura necessários para acompanhar o desenvolvimento da planta.

**Figura 1:** Sensor de Temperatura e Umidade DHT11



**Fonte:** FilipeFlop

## 2.2 Sensor de umidade do solo

O sensor de umidade do solo modelo HD-38 (Figura 2) consiste em duas hastes com dois eletrodos. Aplicando-se uma determinada corrente aos eletrodos, é possível ter um indicador de quanto o solo está úmido. Caso o solo esteja molhado, a condutividade é melhor devido a absorção da água, e caso esteja seco teremos pouca ou nenhuma corrente (ALMEIDA, 2017). Esse sensor foi utilizado para medir a umidade do solo e de acordo com os valores obtidos, acionar ou desligar a irrigação.

**Figura 2:** Sensor de Umidade do solo HD-38



**Fonte:** FilipeFlop

## 2.3 Sensor de chuva

O sensor de chuva modelo MH-RD (Figura 3) foi utilizado neste projeto. Este sensor tem como finalidade detectar gotas de chuva. Caso não sejam detectadas gotas de água na superfície da placa, a saída (digital) do sensor se mantém em nível alto e quando o sensor detectar alguma gota de água sobre a superfície, a saída (digital) altera para nível baixo. Assim como o sensor de temperatura e umidade do ar, este sensor é utilizado para obter dados para monitoramento das condições climáticas durante o período de desenvolvimento da cultura.

**Figura 3:** Sensor de chuva MH-RD



**Fonte:** FilipeFlop

## 2.4 Válvula solenóide

A válvula solenóide (Figura 4) é utilizada principalmente em aplicações voltadas a projetos eletrônicos para controle de fluxo de líquidos e gases. É uma válvula eletromecânica controlada que recebe o nome de solenóide pois seu componente principal é uma bobina elétrica com um núcleo ferromagnético móvel, também chamado de êmbolo. Em repouso, esse êmbolo tampa um orifício por onde circula o fluido e quando a corrente circula pela bobina, cria um campo magnético que exerce uma força no êmbolo. Com isso, o êmbolo é puxado em direção ao centro da bobina, fazendo com que o orifício se abra. (SILVEIRA, 2017). Nesse projeto, a válvula solenóide é usada para acionar ou desligar a irrigação da horta.

**Figura 4:** Válvula solenóide para água 127V 180° (3/4 x 3/4)



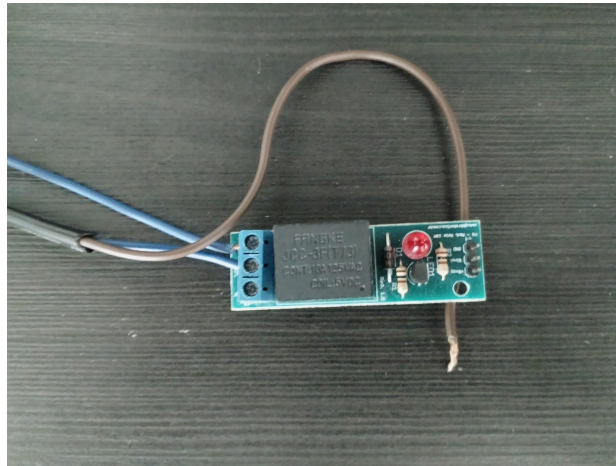
**Fonte:** Baú da Eletrônica



## 2.5 Relé

O relé T73 (Figura 5) é um dispositivo eletromecânico criado para alterar o estado de um circuito elétrico. Ele possui um circuito que aciona o seu eletroímã para mudar a posição dos contatos (MATTEDE, 2023). Neste projeto, o relé foi utilizado para o acionamento da válvula solenóide.

**Figura 5:** Relé T73



**Fonte:** Autores

## 2.6 ESP32

Criado pela Espressif Systems, o ESP32 é um microcontrolador de baixo custo e baixo consumo de energia com *wi-fi* e *bluetooth* integrados de fábrica. Pode ser programado com C++, MicroPython, Lua, Javascript entre outras linguagens (ESP32, 2023). Além das características citadas acima, a escolha por esse microcontrolador se deu em razão da linguagem de programação utilizada no projeto, o MicroPython. Além do mais, o microcontrolador ESP32 possui diversas outras características importantes, tais como ser suficientemente robusto para as características do projeto (opera em ambientes industriais, suporta grandes variações de temperatura), possui as portas necessárias para os sensores necessários, além de possuir uma comunidade muito ativa e atuante na internet.

## 2.7 MicroPython

MicroPython trata-se de uma versão mais enxuta da linguagem de programação Python 3. Possui um pequeno conjunto de biblioteca padrão usadas no Python 3 sendo otimizada para rodar em microcontroladores. (MICROPYTHON, 2022).

## 2.8 Message Queuing Telemetry Transport - MQTT

O protocolo *Message Queuing Telemetry Transport* (MQTT) foi criado pela IBM nos anos 90 com a finalidade de ser um protocolo simples e leve que conseguisse comunicação entre máquinas com baixa latência. O MQTT usa o paradigma de *Publisher/Subscriber*, onde o *Publisher* seria quem publica os dados e o *Subscriber* recebe os dados e faz algo com eles. O servidor MQTT é chamado de *broker* que é responsável em mandar as mensagens publicadas nos pelos *Publishers* para os *Subscribers* baseado no tópico. Outro ponto importante é o *header* das mensagens que são pequenas para otimização de banda (MQTT, 2022). A escolha deste protocolo se deu em razão da facilidade de implementação, ser otimizado para bandas de baixa latência e leve, alinhado com as necessidades desejadas por este trabalho.

## 2.9 Python

Python é uma linguagem de alto nível interpretada com tipagem dinâmica e forte, amplamente usada em aplicações da *Web*, desenvolvimento de *software*, ciência de dados e *machine learning* (ML). Python é eficiente e fácil de aprender e pode ser executado em muitas plataformas diferentes. O Python foi lançado no início da década de 90 pelo programador e matemático holandês Guido Van Rossum. A linguagem foi projetada para dar ênfase no trabalho do desenvolvedor, facilitando a escrita de um código limpo, simples e legível, tanto em aplicações menores quanto em programas mais complexos

(PYTHON, 2022). A escolha do Python se deu para mantermos a mesma linguagem tanto no hardware ESP32, quanto na aplicação *backend*.

## 2.10 Firebase Firestore

*Firebase Firestore* é um banco de dados em nuvem NoSQL flexível e escalável para armazenar e sincronizar dados para desenvolvimento do lado do servidor e do cliente. É um banco de dados flexível e escalável para desenvolvimento em dispositivos móveis, *Web* e servidores do Firebase. Ele oferece suporte *offline* para dispositivos móveis e *Web* para que você possa criar aplicativos responsivos que funcionem independentemente da latência da rede ou da conectividade com a Internet (FIRESTORE, 2022).

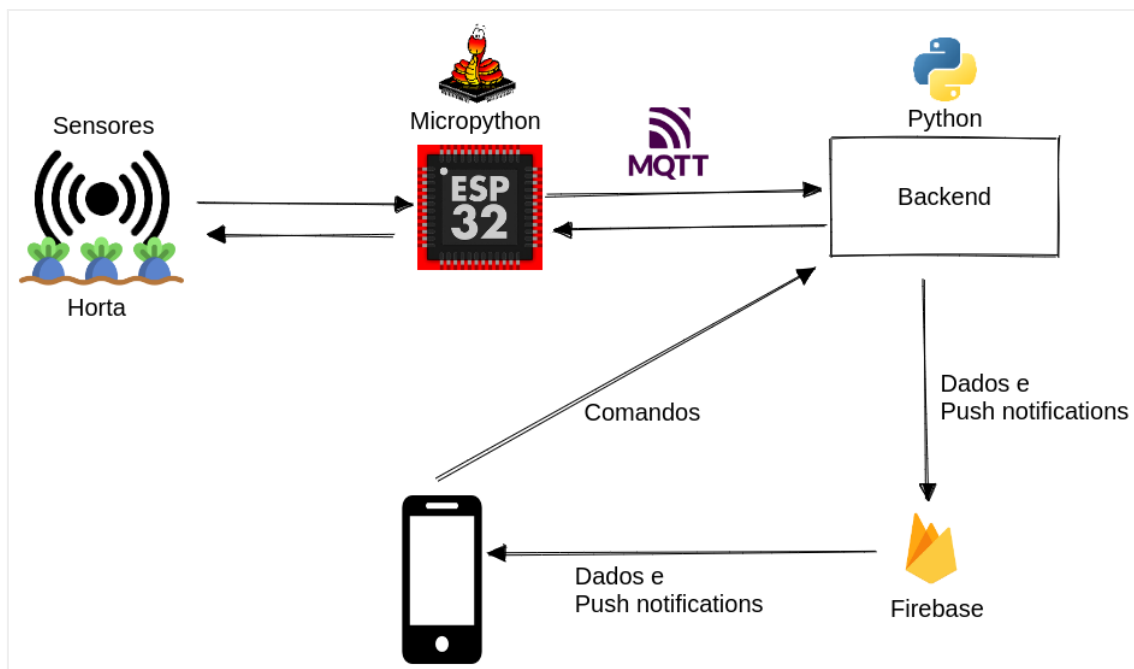
## 2.11 Android

Atualmente desenvolvido pela Google, o Android trata-se de um sistema operacional baseado no Kernel do Linux, lançado em 2008, presente em celulares, tablets, TVs e até mesmo carros (ANDROID, 2022). Neste projeto será implementado um aplicativo para o sistema operacional Android, responsável por apresentar os dados obtidos pelos sensores, em um processo de monitoramento, bem como apresentar uma interface para configuração e controle do processo de irrigação automatizado.

# 3 DESENVOLVIMENTO

Este tópico apresenta o processo de planejamento, projeto e implementação de um sistema automatizado para irrigação de hortas. Também são apresentados o escopo e arquitetura (Figura 6) do projeto, detalhando características relacionadas ao desenvolvimento da aplicação. Ainda serão apresentados o processo de desenvolvimento e principais algoritmos implementados para o ESP32, sistema *backend* e para o aplicativo móvel.

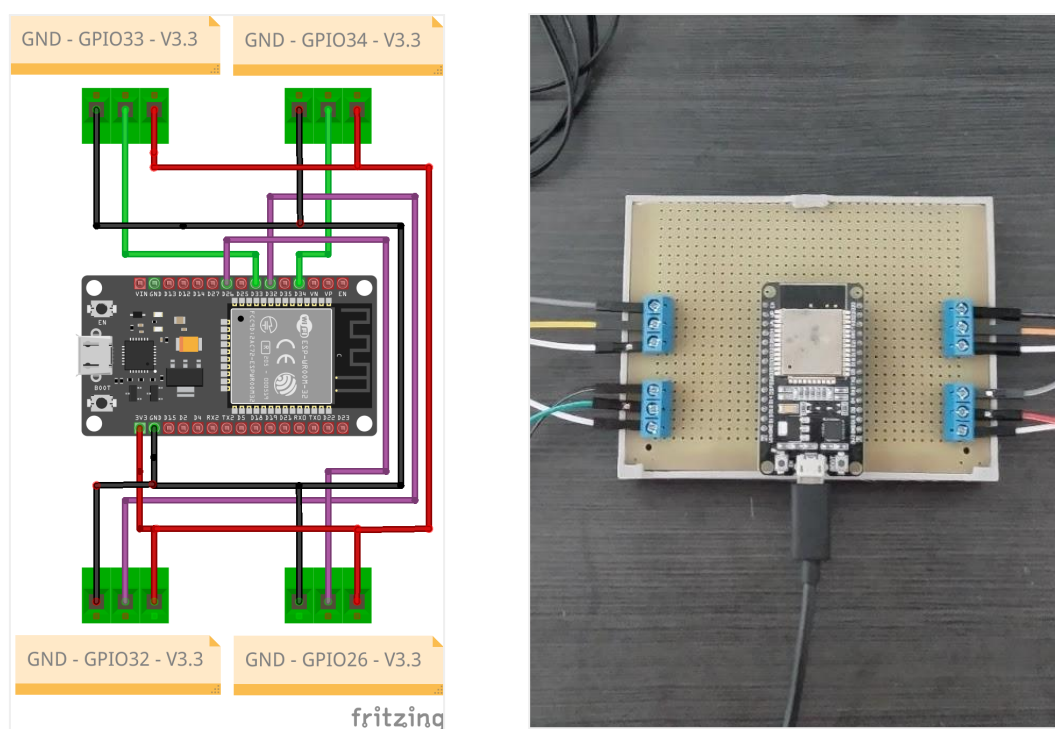
**Figura 6:** Arquitetura projeto



**Fonte:** Autores

Após a definição da arquitetura do sistema e da seleção dos elementos necessários para a elaboração do projeto, iniciou-se o desenvolvimento da parte física do sistema, representada na Figura 6 pelos sensores e pelo ESP32 com MicroPython. Inicialmente, foram adicionados sensores de umidade do solo, temperatura e umidade do ar e o sensor de chuva ao ESP32. Ainda, foram adicionados os atuadores do tipo relé, com o objetivo de acionar ou desligar a válvula solenóide, responsável pela passagem de água e, consequentemente, pela irrigação. A fim de melhor acomodar e proteger o hardware, foi desenvolvido um case, impresso em 3D. Além disso, com o intuito de melhor organizar os componentes, foi confeccionada uma placa em fenolite, com encaixes para suportar o ESP32 e conexões do tipo borne para melhor conexão dos cabos dos sensores. A Figura 7 apresenta o diagrama esquemático do projeto de *hardware* e ao lado, a construção realizada.

**Figura 7:** Diagrama esquemático do projeto



**Fonte:** Autores

De acordo com o projeto apresentado, o sensor de umidade do solo está conectado na porta 34, o sensor de temperatura e umidade do ar na porta 33, o sensor de chuva na porta 32 e o relé na porta 26.

### 3.1 Tópicos MQTT usados

Os dados recebidos pelos sensores, realizados em intervalos de trinta segundos, passaram a ser enviados para o microcontrolador ESP32, executando uma etapa de pré-normalização, enviando-os em seguida para um broker MQTT. Para melhor orquestrar o envio e a recepção de tais dados, as informações de cada um dos sensores e dos atuadores foi organizada em tópicos de interesse, conforme mostrados na Tabela 1.

Tabela 1. Tabela dos tópicos usados na aplicação.

Tópico	Função
ifpr/soil_sensor	Responsável pelo envio dos dados do sensor de umidade do solo para o <i>broker</i> .
ifpr/temp_sensor	Responsável pelo envio dos dados do sensor de temperatura do ar para o <i>broker</i> .
ifpr/humidity_sensor	Responsável pelo envio dos dados do sensor de umidade do ar para o <i>broker</i> .
ifpr/rain_sensor	Responsável pelo envio dos dados do sensor de chuva para o <i>broker</i> .
ifpr/water_control	Responsável em receber os comandos do <i>backend</i> e do aplicativo móvel para ligar ou desligar a irrigação.

Fonte: Autores.

Uma vez enviados para o *broker*, os dados eram passíveis de serem consumidos pela aplicação *backend*, persistindo-os na nuvem, ou realizando outras operações.

### 3.2 Configuração do ESP32 para MicroPython e código

Após realizada conexão dos sensores, o passo seguinte foi a instalação e a configuração do MicroPython no ESP32<sup>1</sup>. Por não se tratar de um processo trivial, o processo é apresentado aqui.

De posse do arquivo binário do MicroPython, para prosseguir com a configuração, foi necessário a instalação da ferramenta ESPTool<sup>2</sup>, cujo objetivo é a de formatar e gravar o MicroPython no microcontrolador. A instalação dessa ferramenta pode ser realizada por meio do comando `pip install esptool`.

Uma vez instalado, o processo de formatação do ESP32 e instalação do MicroPython pode ser realizado realizando-se os seguintes comandos:

<sup>1</sup> O download do MicroPython pode ser encontrado em: <<https://micropython.org/download/>>.

<sup>2</sup> Download do esptool: <<https://github.com/espressif/esptool/>>.

- Para formatar: `esptool --port /dev/ttyUSB0 erase_flash`  
onde o `--port` refere-se a porta USB ligada ao ESP32.
- Para gravar o MicroPython no microcontrolador: `esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000 <binario-micropython>.bin.`

Após instalado o MicroPython no microcontrolador, foi realizada a implementação do código na linguagem citada anteriormente. O MicroPython necessita, essencialmente, de 2 arquivos: o `main.py` e o `boot.py`.

O arquivo `boot.py` é executado quando o microcontrolador realiza o processo de *boot*<sup>3</sup> e o `main.py` é executado na sequência. É no arquivo `main.py` que código fonte para controle do ESP32 será executado. Para transferir os arquivos do computador para o ESP32 utilizou-se a ferramenta Ampy<sup>4</sup>, que pode ser instalada por meio do comando: `pip install adafruit-ampy`. Após instalado a Ampy, é possível usar o comando para transferir os arquivos da seguinte forma: `ampy --port /dev/ttyUSB0 put boot.py` e `ampy --port /dev/ttyUSB0 put main.py`. O arquivo `main.py` ainda pode importar outros arquivos, que por sua vez também precisam ser transferidos para o microcontrolador.

Uma vez realizada as configurações supracitadas, e realizado o *boot*, o código fonte contido no arquivo `main.py` é executado. A Figura 8 apresenta uma visão geral do algoritmo escrito nesse arquivo.

<sup>3</sup> É o processo de inicialização do dispositivo

<sup>4</sup> Download do ampy: <https://github.com/scientifichackers/ampy>

**Figura 8:** Código fonte main.py no ESP32

```
1 from time import sleep
2
3 import rain_sensor
4 import server
5 import soil_sensor
6 import temp_sensor
7 import umqtt_client
8 import wifi
9 import relay
10 from config import topic_water
11
12 is_connected = wifi.connect()
13 publish_time = 30
14
15 def on_message(topic, msg):
16     if msg == b"off":
17         relay.status(0)
18     else:
19         relay.status(1)
20
21
22 if not is_connected:
23     print("Wifi not connected...")
24     wifi.access_point()
25     server.listen()
26 else:
27     print("Wifi connected...")
28     from ntp_time import get_ntp_time
29     client = umqtt_client.init()
30     client.set_callback(on_message)
31     umqtt_client.subscribe([topic_water])
32     count = -1
33     while True:
34         if count == -1 or count == publish_time:
35             timestamp = get_ntp_time()
36             print('Timestamp:', str(timestamp))
37             temp_sensor.publish_mqtt(timestamp)
38             soil_sensor.publish_mqtt(timestamp)
39             rain_sensor.publish_mqtt(timestamp)
40             print('-----')
41             count = 0
42             umqtt_client.check_msg()
43             sleep(1)
44             count += 1
45
46     umqtt_client.disconnect()
```

**Fonte:** Autores

De forma geral, o *script* se inicia por meio da verificação de existência de uma conexão do ESP32 à uma rede *wi-fi*. Caso esteja conectado e não consiga proceder com a conexão, é criado um *access point*<sup>5</sup> nomeado de “automated-garden” e cuja senha é “esp1234567890”. Após conectar-se nesse *access point*, basta acessar o endereço `http://192.168.4.1/` via navegador,

<sup>5</sup> Uma rede *wi-fi* criada pelo ESP para que outros dispositivos conectem nele



selecionando a rede mencionada e informando a senha. Novamente o ESP32 é reinicializado e tenta conectar-se à rede *wi-fi* recém configurada. Caso obtenha sucesso na conexão, é iniciado o fluxo de criação do cliente *umqtt*. No cliente configurou-se o método *on\_message* como responsável em receber as mensagens MQTT, subscreve-se no tópico *ifpr/water\_control* para o recebimento do comando de acionamento da irrigação e começa a publicação dos dados dos sensores de temperatura e umidade do ar, umidade do solo e chuva para seus respectivos tópicos citados anteriormente. Ressalta-se que o tempo de envio dos dados dos sensores para o *backend* é de trinta segundos.

### 3.3 Aplicativo Android

Com o intuito de permitir ao usuário controlar, monitorar a irrigação e acompanhar os dados obtidos por meio dos sensores, foi implementado um aplicativo para o sistema operacional Android. Além disso, o aplicativo permite a configuração dos horários de irrigação, o indicador de umidade desejado, bem como permite acionar ou parar a irrigação a qualquer momento.

**Figura 9:** Tela inicial aplicativo Android



**Fonte:** Autores

A tela inicial do aplicativo apresenta um componente do tipo “switch” cujo intuito é mostrar se a irrigação está ligada ou desligada. Este componente permite o acionamento manual do processo de irrigação (Figura 9). Abaixo desse componente exibiu-se os horários para início da irrigação e o indicador de umidade do solo desejado para encerrar esse processo que foram obtidos da coleção chamada *presets* no *Firebase Firestore*.

O algoritmo implementado no aplicativo, realiza a conexão com *broker* MQTT e inscreve-se nos tópicos: *ifpr/soil\_sensor*, *ifpr/temp\_sensor*, *ifpr/humidity\_sensor*, *ifpr/rain\_sensor* para receber os valores obtidos a partir da leitura dos sensores e *ifpr/water\_control* para receber o comando de ligar e desligar a irrigação. Ao receber esses valores dos sensores via MQTT, são mostrados os dados obtidos em seus respectivos componentes.

Ainda, os dados dos últimos trinta dias, das coleções *ifpr\_soil\_sensor*, *ifpr\_temp\_sensor*, *ifpr\_humidity\_sensor* e *ifpr\_rain\_sensor* armazenados no *Firebase Firestore*, são recuperados e plotados em gráficos a fim de melhor apresentar a temporalidade desses dados.

Uma outra funcionalidade importante presente no aplicativo, é a possibilidade de edição dos horários para início da irrigação e o da umidade do solo para seu encerramento.

Essa funcionalidade é apresentada em uma nova tela. Há a possibilidade de adicionar até dois horários além da umidade do solo ideal, momento em que será desligada a água. Esses dados são armazenados no *Firebase Firestore* na coleção chamada “*presets*”. A Figura 10 apresenta a tela para cadastro das configurações de irrigação..

**Figura 10:** Tela para adicionar os dados de irrigação

The screenshot shows a mobile application interface titled 'Automated Garden'. It features three input fields: 'Horário 1', 'Horário 2', and 'Umidade do solo para desligar a irrigação'. Below these fields is a red button labeled 'ADICIONAR'. The interface is displayed on a smartphone screen with a status bar at the top showing the time 3:49 and various icons, and an Android navigation bar at the bottom.

**Fonte:** Autores

Além de receber os dados dos sensores, verificar o estado do processo de irrigação via MQTT e exibir as informações ao usuário, o aplicativo é responsável também em mostrar as notificações enviadas pelo *backend* quando a irrigação é iniciada ou encerrada.

### 3.4 Backend (Python)

A aplicação *backend*, escrita em linguagem Python, é responsável por receber os dados dos sensores, enviados via MQTT pelo microcontrolador ESP32. Após receber esses dados, o *backend* armazena esses dados na base de dados em nuvem *Firebase Firestore*, em coleções específicas para cada sensor. Também é responsabilidade dessa aplicação verificar os horários para iniciar o fluxo de água e verificar a umidade do solo, para proceder com o

encerramento da irrigação e envio da notificação para o dispositivo Android, que ocorre sempre que o estado do processo de irrigação é modificado.

O algoritmo implementado para essa aplicação, conecta-se ao *broker* MQTT, inscreve-se nos tópicos *ifpr/soil\_sensor*, *ifpr/temp\_sensor*, *ifpr/humidity\_sensor*, *ifpr/rain\_sensor* para então, receber os dados dos respectivos sensores. A função *on\_message* trata do recebimento desses dados, conforme pode ser observado na Figura 11.

**Figura 11:** Código da função *on\_message*

```
1 def on_message(client, userdata, msg):
2     value = msg.payload.decode()
3     data = json.loads(value)
4
5     set_value(msg.topic, data['value'])
6     save_value(msg.topic, data)
7
8     if msg.topic == topic_soil.decode():
9         check_water_status(data['timestamp'])
```

**Fonte:** Autores

Após receber esses dados, eles são persistidos no *Firebase Firestore* nas coleções previamente citadas, realizando-se a verificação da lógica para acionamento da irrigação programada pela função *check\_water\_status*.

Nesta função, são obtidos os dados das configurações de irrigação, persistidas na coleção “*presets*” do banco de dados, e em seguida é realizada a validação do horário e a umidade do solo (Figura 12).

**Figura 12:** Lógica de acionamento da irrigação

```
1 if soil_value >= int(soil_humidity):
2     if water_status:
3         send_water_status('off', timestamp)
4         save_water_status('off', timestamp)
5         my_firebase.send_notification(firestore_client, "Automated Garden", "Irrigação
6         desligada com sucesso...")
7         water_status = False
8     else:
9         if (can_start_time or can_end_time) and water_status == False:
10            send_water_status('on', timestamp)
11            save_water_status('on', timestamp)
12            my_firebase.send_notification(firestore_client, "Automated Garden", "Irrigação ligada
13            com sucesso...")
14            water_status = True
```

**Fonte:** Autores.

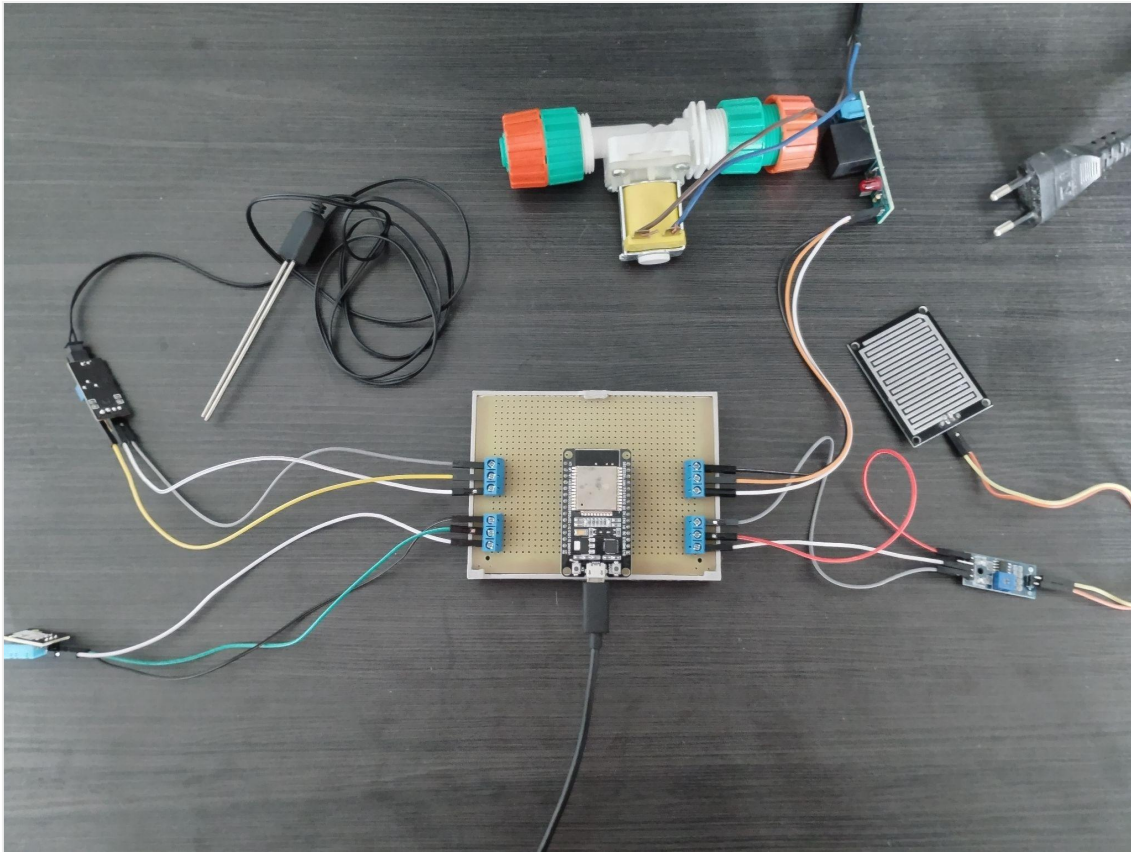
Conforme detalhado no código acima, primeiramente é verificado se a umidade do solo é maior que a informada nas configurações de irrigação. Caso seja maior, é verificada se a irrigação já está acionada e caso esteja, é publicado o valor “off” no tópico `ifpr/water_control`. Os estados da irrigação são salvos na coleção `ifpr_water_control` no *Firebase Firestore* e posteriormente uma notificação é enviada para o dispositivo, informando que a irrigação foi desligada.

Já para os casos em que a umidade do solo seja menor do que o informado nas configurações de irrigação, verifica-se se o horário 1 ou horário 2 são iguais ao horário atual e o estado da água esteja desligado. Caso essa lógica seja verdadeira, é publicado o valor “on” no tópico `ifpr/water_control`, salvando o estado da irrigação na coleção `ifpr_water_control` no *Firebase Firestore* e enviando uma notificação para o aplicativo móvel, notificando o início da irrigação.

## 4 RESULTADOS

Considerando o exposto nos tópicos anteriores, este tópico tem por objetivo apresentar o funcionamento da solução implementada, por meio de uma prova de conceito, visando demonstrar suas principais características, funcionalidades e os benefícios do monitoramento e da irrigação automatizada. Assim, a construção do *hardware* pode ser observada na Figura 13.

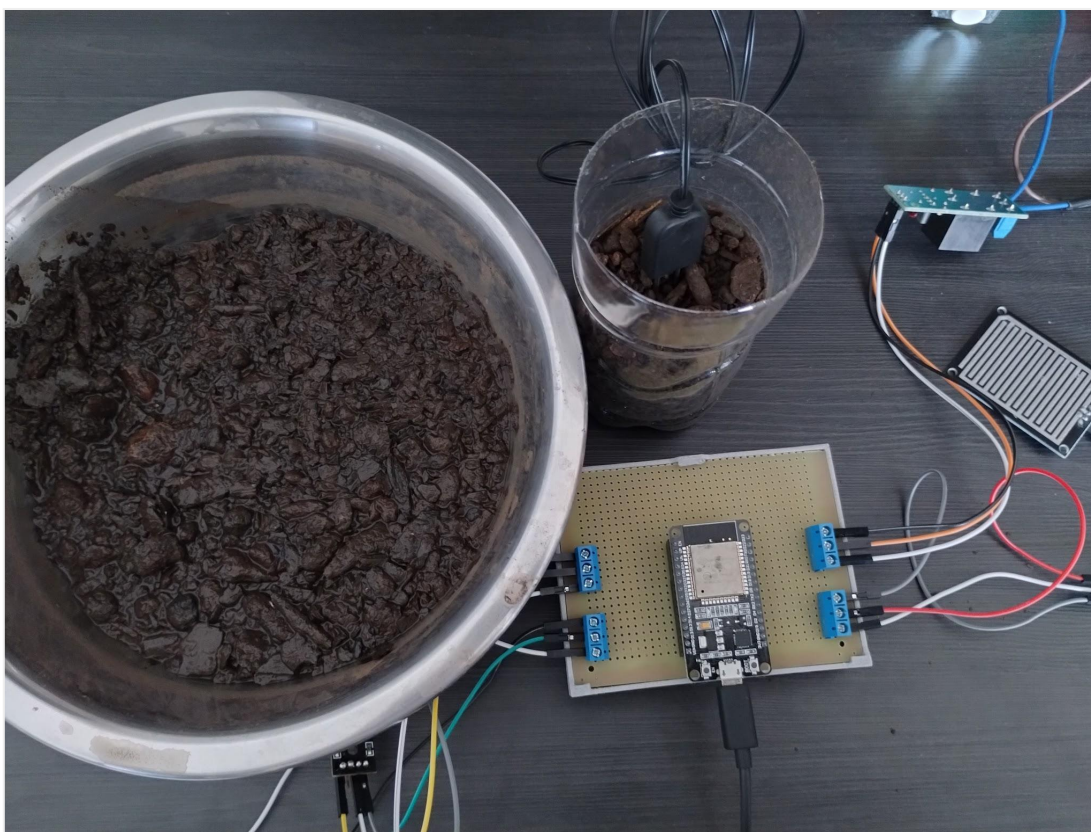
**Figura 13:** Dispositivo com os sensores, relé e a válvula solenóide



**Fonte:** Autores

Para realização da prova de conceito, foram utilizados dois recipientes, um com terra seca e outro com terra molhada, conforme apresentado pela Figura 14. Foi ligado o dispositivo, acomodando o sensor de umidade do solo dentro de um recipiente com terra seca com cerca de 8 centímetros de profundidade.

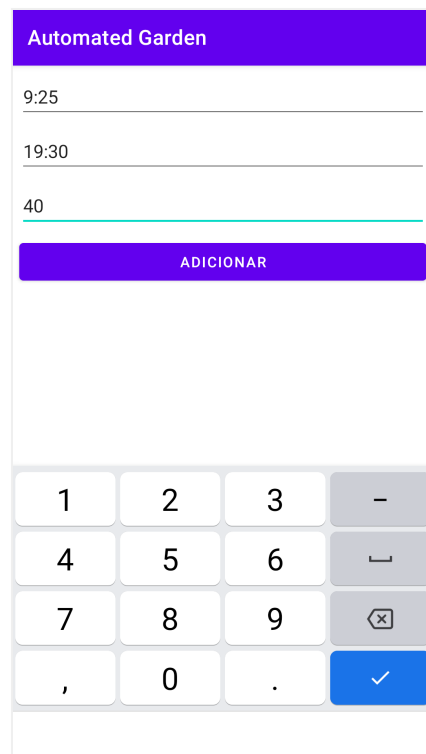
**Figura 14:** Dispositivo e recipientes com terra



**Fonte:** Autores

Após ligar o dispositivo e colocar o sensor de umidade do solo na terra seca, foi realizada a configuração de irrigação para o funcionamento com horários pré-definidos para às nove horas e vinte e cinco minutos e às dezenove horas e trinta minutos, com o desligamento configurado para quando a umidade do solo alcançar 40% ou mais. A Figura 15 apresenta a tela do aplicativo com essa configuração.

**Figura 15:** Tela inicial do aplicativo



Fonte: Autores

Ao ser ligado, assim como já mencionado no tópico anterior, caso o microcontrolador ESP32 não seja capaz de realizar uma conexão com uma rede *wi-fi* configurada, o *access point* previamente configurado é criado, conforme apresentado na Figura 16.

**Figura 16:** Log Micropython do microntrolador ESP32

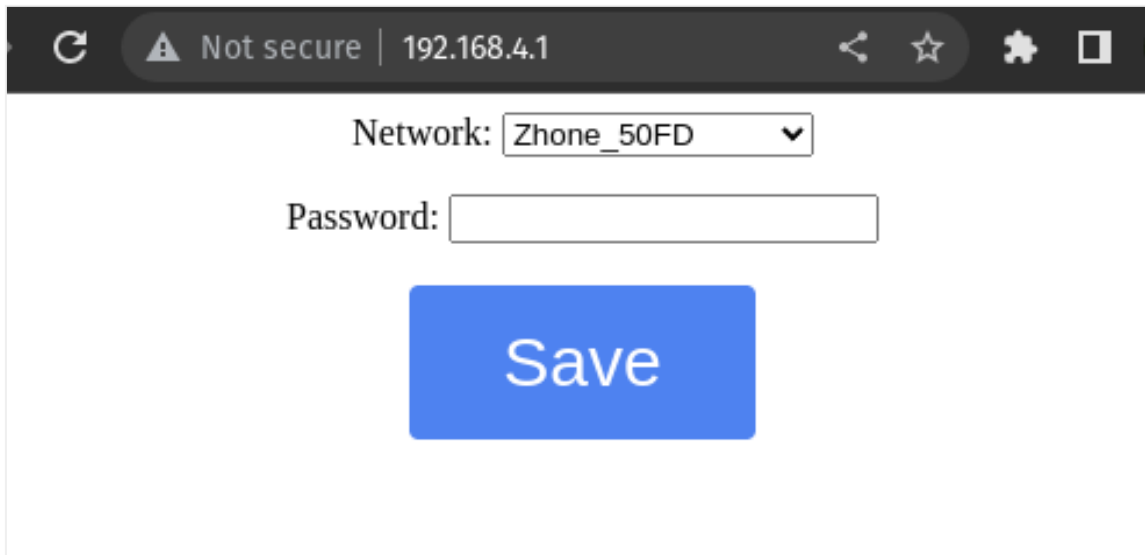
```
Shell
ho 0 tail 12 room 4
load:0x40078000,len:12344
ho 0 tail 12 room 4
load:0x40080400,len:4124
entry 0x40080680
['1', '2\n']
Connecting to network 1.....
Wifi network config: ('0.0.0.0', '0.0.0.0', '
0.0.0.0', '0.0.0.0')
Wifi not connected...
Access Point active: True
Connect to automated-garden network with pas
sword esp1234567890 and access: http://192.168.4.1
in you browser
```

Fonte: Autores



Assim, realizou-se a conexão com esse *access point*, acessado por meio do endereço `http://192.168.4.1`, configurando assim a rede *wi-fi* para o dispositivo (Figura 17).

**Figura 17:** Acesso ao access point para configurar o *wi-fi*



Fonte: Autores

Após o processo de configuração da rede *wi-fi*, é realizado automaticamente o *reboot* do sistema. A partir desse momento o ESP32 inicia o processo de obtenção dos dados dos sensores, enviando-os para os tópicos mencionados anteriormente, configurados no *broker* MQTT, (Figura 18).

**Figura 18:** ESP32 conectado a *wi-fi* e envio dos dados dos sensores



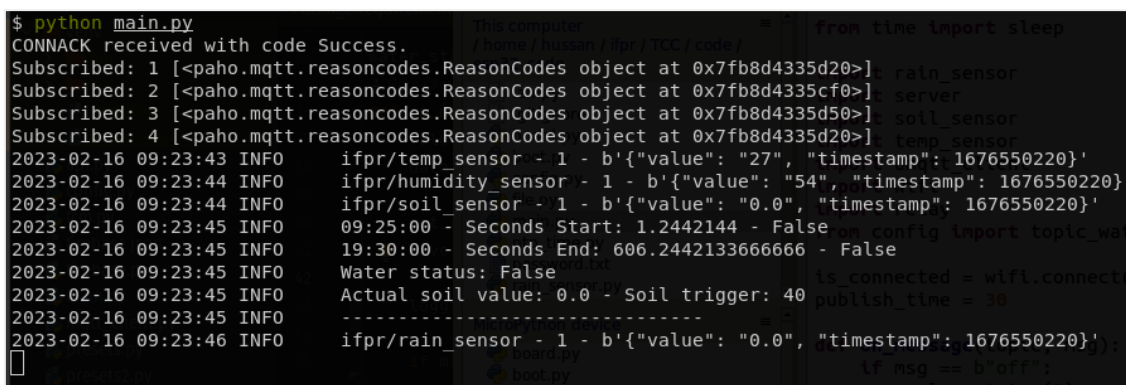
```
Connecting to network Zhone_50FD.....
Wifi network config: ('192.168.1.23', '255.255.255.0', '192.168.1.1', '192.168.1.1')
Wifi connected...
ESP is Connected to broker.hivemq.com in port 1883 and subscribed to b'ifpr/water_control' topic
Timestamp: 1676473689
Temperature: 23.00°
Air Humidity: 55.00%
Soil Humidity: 0.00% (adc: 1023)
Rain sensor: 0.00% (adc: 1023)
-----
```

Fonte: Autores

Os dados enviados para o broker MQTT são obtidos pela aplicação *backend*, que, assim que inicializada, se inscreve nos tópicos dos sensores e começa a monitorar a chegada das informações nesses tópicos. Tais informações são utilizadas pela aplicação e em seguida são persistidas no banco de dados em nuvem.

A aplicação *backend* ainda, em intervalos de tempo, verifica as configurações definidas para a irrigação, a fim de acionar a válvula para abertura da água. A Figura 19 apresenta o processo de inscrição nos tópicos além de um extrato das informações coletadas pelos sensores.

**Figura 19:** Inscrição nos tópicos e recebimento dos dados.



```

$ python main.py
CONNACK received with code Success.
Subscribed: 1 [<paho.mqtt.reasoncodes.ReasonCodes object at 0x7fb8d4335d20>] rain_sensor
Subscribed: 2 [<paho.mqtt.reasoncodes.ReasonCodes object at 0x7fb8d4335cf0>] server
Subscribed: 3 [<paho.mqtt.reasoncodes.ReasonCodes object at 0x7fb8d4335d50>] soil_sensor
Subscribed: 4 [<paho.mqtt.reasoncodes.ReasonCodes object at 0x7fb8d4335d20>] temp_sensor
2023-02-16 09:23:43 INFO ifpr/temp_sensor - 1 - b'{"value": "27", "timestamp": 1676550220}'
2023-02-16 09:23:44 INFO ifpr/humidity_sensor - 1 - b'{"value": "54", "timestamp": 1676550220}'
2023-02-16 09:23:44 INFO ifpr/soil_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550220}'
2023-02-16 09:23:45 INFO 09:25:00 - Seconds Start: 1.2442144 - False
2023-02-16 09:23:45 INFO 19:30:00 - Seconds End: 606.2442133666666 - False
2023-02-16 09:23:45 INFO Water status: False
2023-02-16 09:23:45 INFO Actual soil value: 0.0 - Soil trigger: 40
2023-02-16 09:23:45 INFO -----
2023-02-16 09:23:46 INFO ifpr/rain_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550220}'}

```

**Fonte:** Autores

Para este caso de uso, foi verificado que quando o horário configurado e a umidade do solo atingem valor igual ou maior do que o indicado na configuração, a válvula solenóide é acionada pelo microcontrolador ESP32, com o envio valor “on” para o tópico `ifpr/water_control`, previamente citado, e ao atingir a umidade do solo desejado para o desligamento, foi enviado um valor “off” para o mesmo tópico (Figura 20). Nas duas situações, notificações foram enviadas para o aplicativo móvel.

**Figura 20:** Acionamento da água e envio de mensagem pro aplicativo móvel

```

2023-02-16 09:24:51 INFO -----
2023-02-16 09:24:52 INFO ifpr/rain_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550287}'
2023-02-16 09:25:22 INFO ifpr/temp_sensor - 1 - b'{"value": "27", "timestamp": 1676550320}'
2023-02-16 09:25:23 INFO ifpr/humidity_sensor - 1 - b'{"value": "54", "timestamp": 1676550320}'
2023-02-16 09:25:24 INFO ifpr/soil_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550320}'
2023-02-16 09:25:25 INFO 09:25:00 - Seconds Start: -0.4222140000000003 - True
2023-02-16 09:25:25 INFO 19:30:00 - Seconds End: 604.5777851666667 - False
2023-02-16 09:25:25 INFO Water status: False
2023-02-16 09:25:25 INFO Actual soil value: 0.0 - Soil trigger: 40
2023-02-16 09:25:25 INFO -----
2023-02-16 09:25:25 INFO Water State - on - 1676550320
1 messages were sent successfully
mid: 5
2023-02-16 09:25:28 INFO ifpr/rain_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550320}'
2023-02-16 09:25:54 INFO ifpr/temp_sensor - 1 - b'{"value": "27", "timestamp": 1676550353}'
2023-02-16 09:25:55 INFO ifpr/humidity_sensor - 1 - b'{"value": "54", "timestamp": 1676550353}'
2023-02-16 09:25:56 INFO ifpr/soil_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550353}'
2023-02-16 09:25:57 INFO 09:25:00 - Seconds Start: -0.9509104333333334 - True
2023-02-16 09:25:57 INFO 19:30:00 - Seconds End: 604.0490887333333 - False
2023-02-16 09:25:57 INFO Water status: True
2023-02-16 09:25:57 INFO Actual soil value: 0.0 - Soil trigger: 40
2023-02-16 09:25:57 INFO -----
2023-02-16 09:25:57 INFO ifpr/rain_sensor - 1 - b'{"value": "0.0", "timestamp": 1676550353}'
2023-02-16 09:26:28 INFO ifpr/temp_sensor - 1 - b'{"value": "27", "timestamp": 1676550386}'
2023-02-16 09:26:29 INFO ifpr/humidity_sensor - 1 - b'{"value": "54", "timestamp": 1676550386}'
2023-02-16 09:26:30 INFO ifpr/soil_sensor - 1 - b'{"value": "73.60704", "timestamp": 1676550386}'
2023-02-16 09:26:30 INFO 09:25:00 - Seconds Start: -1.5160974833333334 - False
2023-02-16 09:26:30 INFO 19:30:00 - Seconds End: 603.4839017166667 - False
2023-02-16 09:26:30 INFO Water status: True
2023-02-16 09:26:30 INFO Actual soil value: 73.60704 - Soil trigger: 40
2023-02-16 09:26:30 INFO -----
2023-02-16 09:26:30 INFO Water State - off - 1676550386
1 messages were sent successfully
mid: 6

```

Fonte: Autores

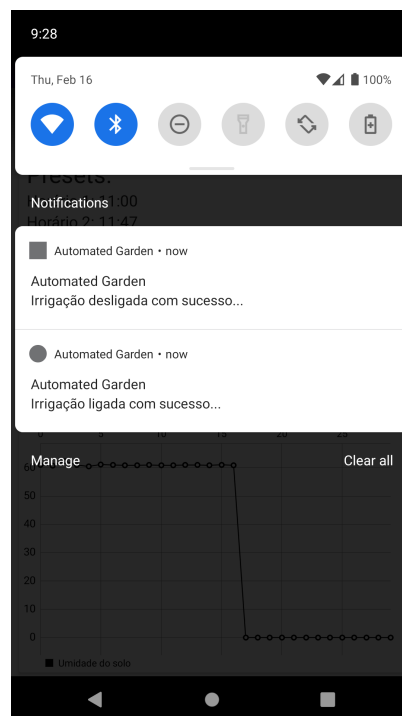
Para monitoramento e demais configurações, é usado o aplicativo Android. Ao ser aberto, são exibidos informações da configuração realizada anteriormente. Ainda, foram apresentados gráficos das últimas trinta medições, obtidos a partir das informações coletadas pelos sensores, e armazenadas no *Firebase Firestore*. A irrigação foi ligada quando a lógica de acionamento foi verdadeira, sendo também enviada uma notificação para o usuário. Adicionalmente, quando o solo atingiu a umidade configurada, recebeu-se uma notificação de desligamento (Figuras 21 e 22).

**Figura 21:** Acionamento da água e dados nos gráficos



**Fonte:** Autores

**Figura 22:** Recebimento da mensagem no aplicativo móvel



**Fonte:** Autores

## 5 CONSIDERAÇÕES FINAIS

O processo de irrigação é essencial para a saúde das plantas e aumento da produtividade das hortas, contudo gerenciar tal atividade não é uma tarefa trivial, uma vez que exige mão de obra qualificada, tempo e pode aumentar o desperdício de recursos, quando realizado de maneira incorreta. Nesse sentido, este trabalho apresenta uma prova de conceito para um sistema de irrigação automatizado e com possibilidade de monitoramento.

O projeto, testado em bancada, apresentou solução satisfatória para o problema apresentado. O trabalho focou-se na resolução do problema, de modo que possíveis melhorias podem ser realizadas por meio de trabalhos futuros, tais como: uma arquitetura que atenda a mais de um usuário e um dispositivo, a possibilidade de adicionar mais de uma configuração de irrigação, configurar mais de 2 horários, e a possibilidade de autenticação do usuário.

## REFERÊNCIAS

AL-FUQAHA, A. et al. **Internet of things: A survey on enabling technologies, protocols, and applications**. IEEE Communications Surveys & Tutorials, v. 17, n. 4, p. 2347-2376, 2015

ALMEIDA, Danilo - **Sensor de umidade do solo com Arduino – Higrômetro**. 2017 - Disponível em <<https://portal.vidadesilicio.com.br/sensor-de-umidade-do-solo-higrometro/>> - Acessado em Janeiro 2023

ANDROID - **Android | A plataforma que redefine o impossível** - Disponível em <[https://www.android.com/intl/pt-BR\\_br/](https://www.android.com/intl/pt-BR_br/)> - Acessado em Dezembro 2022

BARBOSA, José Wilian. **Sistema de Irrigação Automatizado utilizando Arduino**. Assis – SP. 2013 - Disponível em: <<https://cepein.femanet.com.br/BDigital/arqTccs/1011330043.pdf>> - Acessado em: 07 de Fevereiro de 2023

DRESCH, Aline; LACERDA, Daniel Pacheco; JUNIOR, Jose Antonio Valle Antunes. **Design Science Research: Método de Pesquisa para Avanço da Ciência e Tecnologia**. Porto Alegre: Bookman, 2015

UEMG - **Perspectiva na Horticultura**. 2018 - Disponível em  
<[https://editora.uemg.br/images/livros-pdf/catalogo-2021/Perspectivas/perspectivas\\_na\\_horticultura.pdf](https://editora.uemg.br/images/livros-pdf/catalogo-2021/Perspectivas/perspectivas_na_horticultura.pdf)> - Acessado em Janeiro 2023

ESP32 - **ESP32 Wi-Fi & Bluetooth MCU | Espressif Systems** - Disponível em  
<<https://www.espressif.com/en/products/socs/esp32>> - Acessado em Janeiro 2023

FIRESTORE - **Firestore | Firebase** - Disponível em  
<<https://firebase.google.com/docs/firestore>> - Acessado em Dezembro 2022

GUIMARÃES, Vinícius Galvão. **Automação e monitoramento remoto de sistema de irrigação na agricultura**. Brasília, 2011 - Disponível em:  
<[www.ene.unb.br/adolfo/Monographs/Graduation/TG11%20Vinicius%20Galvao.pdf](http://www.ene.unb.br/adolfo/Monographs/Graduation/TG11%20Vinicius%20Galvao.pdf)> - Acessado em 05 de Fevereiro de 2023

JORGE, Marçal Henrique Amici; et al. **Implantação e condução de uma horta de médio porte**. Circular técnica. EMBRAPA. ISSN 1415-3033. Brasília – DF. 2016 - Disponível em:  
<<https://www.embrapa.br/hortalicas/busca-de-publicacoes/-/publicacao/1063739/implantacao-e-conducao-de-uma-horta-de-medio-porte>> - Acessado em Janeiro de 2023

MATTEDE, Henrique - **Relé – Funcionamento, tipos e aplicações!** - Disponível em:  
<<https://www.mundodaeletrica.com.br/rele-funcionamento-tipos-aplicacoes/>> - Acessado em Janeiro 2022

MICROPYTHON - **MicroPython - Python for microcontrollers** - Disponível em: <<https://micropython.org/>> - Acessado em Novembro 2022

MURTA, José - **Sensores DHT11 e DHT22: Guia básico - Blog da Eletrogate**. 2019 - Disponível em  
<<https://blog.eletrogate.com/sensores-dht11-dht22/>> - Acessado em Janeiro de 2023

MARQUELLI, Waldir A; SILVA, Washington L. C. **Seleção de sistemas de irrigação para Hortaliças**. Circular técnica. EMBRAPA. ISSN 1415-3033. Brasília – DF. 2011 - Disponível em:<  
<https://ainfo.cnptia.embrapa.br/digital/bitstream/item/75698/1/ct-98.pdf>> - Acessado em Janeiro de 2023.

MQTT - **The Standard for IoT Messaging** - Disponível em <<https://mqtt.org/>> - Acessado em Novembro 2022

PEFFERS, Ken et. al. **A design science research methodology for information systems research**. Journal of Management Information Systems, v. 24, n. 3, p. 4577, 2007 - Disponível em:

**Revista Mundi Engenharia, Tecnologia e Gestão**. Paranaguá, PR, v.??, n.??, p. ??-??, ??/??., 20??  
DOI: 10.21575/xxxxxxxxxxxxxxxxxxxxxxxxxx

<<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.535.7773&rep=rep1&type=pdf>>. - Acessado em 05 de Fevereiro de 2023

PYTHON - **Welcome to Python.org** - Disponível em <<https://www.python.org/>>  
- Acessado em Dezembro 2022

TESTEZLAF, Roberto. **Irrigação: Métodos, Sistemas e Aplicações**.  
Campinas – SP. 2017 - Disponível em:  
<<http://www.bibliotecadigital.unicamp.br/document/?down=74329>> - Acessado  
em 08 de Fevereiro de 2023