Paulo Vicktor Felix Dias

Sistema Automático de Irrigação e Controle de Temperatura para Estufas

UBERLÂNDIA

2021

Paulo Vicktor Felix Dias

Sistema Automático de Irrigação e Controle de Temperatura para Estufas

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU Faculdade

de Engenharia Elétrica

Orientador: Prof. Dr. Renato Santos Carrijo

UBERLÂNDIA

2021

Dias, Paulo Vicktor Felix

Sistema Automático de Irrigação e Controle de Temperatura para Estufas/ **Paulo Vicktor Felix Dias**. – **UBERLÂNDIA**, **2021**- 28 p. 30 cm.

Orientador: Prof. Dr. Renato Santos Carrijo

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia – UFU. Faculdade de Engenharia Elétrica. **2021**.

Inclui bibliografia.

1. Automação de Hortas. 2. Protocolo MQTT. 3. Irrigação. I. Renato Santos Carrijo. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

Aqui deverá ser colocada na versão final a folha de aprovação da banca.

Agradecimentos

Agradeço a sociedade brasileira por estruturar e manter o ensino público de qualidade. Aos meus familiares que se empenharam em me dar as condições para exercer meus estudos. Aos meus colegas que tornaram o trajeto mais agradável, e a todos os meus amigos por sempre me darem motivação e esperança durante toda a jornada. Ao professor orientador Renato Santos Carrijo, pela paciência, disponibilidade e instrução no decorrer do projeto, e também a ele e a todos os outros professores, o meu agradecimento pelas instruções dadas e os conhecimentos compartilhados ao longo do curso.

“Somos feitos de poeira de
estrelas. Nós somos uma maneira de o cosmos
se autoconhecer.”,
(Carl Sagan)

Resumo

Com foco no plantio realizado em vasos, nas regiões urbanas, propõe-se um modelo de automação do processo de irrigação e de supervisão das variáveis do ambiente, a exemplo: umidade do solo; umidade do ar; temperatura do ar e o volume de água injetado durante a irrigação. Para isso, foi utilizada uma estação de sensoriamento e atuação dessas condições da planta. Sendo elas, controladas por um microcontrolador ESP32 que se comunica via protocolo MQTT com um armazenamento em nuvem, nos servidores da Adafruit IO. Dessa forma, o usuário é capaz de supervisionar os resultados e observar o histórico das condições de seu cultivo. Sistema esse que colhe os valores condicionais e a partir deles atua no ambiente de maneira precisa, otimizando o uso dos recursos e a absorção dos nutrientes pela planta. Ao longo do desenvolvimento, para encontrar as configurações ideais de atuação no sistema, modelos distintos foram testados. Cada modelo apresentava uma atuação diferente, com relação a fatores como a distribuição de diferentes horários e o *set point* referente à umidade do solo. Foi possível concluir que a montagem da arquitetura atingiu os objetivos propostos por esse trabalho e um modelo final foi configurado.

Palavras-chave: Automação de Hortas; Protocolo MQTT; Irrigação.

Abstract

Focusing on planting carried out in pots, in urban areas, a model for the automation of the irrigation process and the supervision of environmental variables is proposed, such as: soil humidity; air humidity; air temperature and the volume of water injected during irrigation. For this, a station for sensing and functioning of these plant conditions was used. They are controlled by an ESP32 microcontroller that communicates via MQTT protocol with a cloud storage, on Adafruit IO servers. In this way, the user is able to supervise the results and observe the history of their cultivation conditions. This system collects the conditional values and acts precisely from them in the environment, optimizing the use of resources and the absorption of nutrients by the plant. During the development, to find out the ideal configurations to act on the system, different models were tested. Each model had a different performance, in relation to factors such as the distribution of different times and the set point referring to soil humidity. It was possible to realize that an assembly of the architecture reached the objectives proposed by this work and a final model was configured.

Palavras-chave: Garden Automation; MQTT protocol; Irrigation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	*Analog to Digital*
AO	*Analog Output*
DO	*Digital Output*
EMBRAPA	*Empresa Brasileira de Pesquisa Agropecuária*
HTTP	*HyperText Transfer Protocol*
IBM	*International Business Machines*
IDE	*Ambiente de Desenvolvimento Integrado*
IHM	*Interface Homem Máquina*
IoT	*Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
LED	*Diodo Emissor de Luz*
MQTT	*Message Queue Telemetry Transport*
NTC	*Negative Temperature Coefficient*
OASIS	*Organization for the Advancement of Structured Information Standards*
QoS	*Quality of Service*
SoC	*System on a Chip*
SSL	*Secure Sockets Layer*
SSR	*Solid State Relay*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UART	*Universal Asynchronous Receiver/Transmitter*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*
USB	*Universal Serial Bus*
WRI	*World Resources Institute*

Sumário

1. Introdução

O Brasil é conhecido mundialmente como um grande exportador de commodities agrícolas, mas não só para exportações são destinadas às plantações Brasileiras, possuindo também modelos de agricultura familiar, nas quais são citadas pelo portal do ministério da agricultura (2020) como principais responsáveis pelo fornecimento de alimentos para a mesa do brasileiro. Esses modelos têm se diversificado ainda mais, atingido novas regiões e até mesmo ambientes urbanos e residenciais, como é o caso da empresa brasileira Pink Farm (2020), a primeira fazenda vertical urbana e comercial da América Latina, onde as plantas são cultivadas em um ambiente totalmente controlado, fechado, e alimentadas por luzes de LED (Diodo Emissor de Luz) azul e rosa, que simulam a luz do sol e aceleram a fotossíntese. A água e o adubo são fornecidos em doses exatas e os agrotóxicos são dispensados nesse sistema.

Conforme pontuado por Lucena (2018), na Revista Ibero-Americana de Ciências Ambientais, todo esse desenvolvimento das atividades de agricultura em direção às regiões urbanas tem como foco subsidiar a segurança alimentar e nutricional das populações incluídas nas cidades, principalmente as com alto grau de densidade demográfica. Assim, ao observar este crescente mercado da automação agrícola, podemos interligar os conceitos tecnológicos da agricultura de precisão com ideias de arquiteturas para domótica e soluções de automação residencial, de forma que os modelos de casas inteligentes possam gerir também o cultivo de hortaliças e vegetais para uma alimentação equilibrada de seus moradores. E com isso, ao aproximar os avanços de um cenário de mercado (agricultura de precisão), com uma realidade que permita que com o auxílio de um algoritmo computacional, qualquer pessoa possa usufruir de uma horta feita em vasos na sua casa, é criado um cenario que contribui para uma popularização do processo de cultivo de plantas, visto que nem todas as pessoas têm conhecimentos acerca de como lidar com as intempéries do ambiente, e ao fornecer essas informações condicionais das plantas, criamos um fator determinante que pode contribuir para uma tomada de decisão correta.

Para isso, pensando em possíveis arquiteturas que podem ser montadas para ser a base do sistemas automático de cultivo, e que também seja capaz de trocar informações com a central de uma casa inteligente, destacasse o protocolo MQTT (Message Queue Telemetry Transport) que encontrou seu espaço nesse universo de conectar objetos via internet, por ser um protocolo simples sem deixar de contemplar características como segurança, qualidade de serviço e pela sua facilidade de implementação.

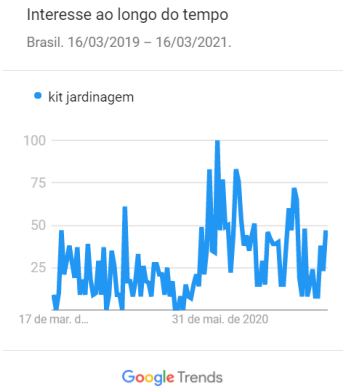
1.1 Justificativa

Para o desenvolvimento de uma planta condições climáticas e ambientais são fatores chaves, que podem favorecer ou prejudicar o cultivo dos vegetais. De acordo com a Agência Embrapa de Informação e Tecnologia (2011) a soja possui condições ideais de cultivo, necessitando ser cultivada entre 20° e 30° C ou terá uma má formação da planta, queda de fatores reprodutivos e diminuição da produção, mas não só a soja possui condições ideais para seu desenvolvimento, cada planta possui suas características e necessidades. Nas pesquisas desenvolvidas nos laboratórios da Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) (2020) nas empresas estaduais de pesquisa e em universidades brasileiras, buscando soluções em termos de adaptação, como por exemplo moldar variantes genéticas de soja, milho, feijão, café, mandioca e algumas frutas para serem mais tolerantes às altas temperaturas e ao déficit hídrico, até mesmo nos estudos mais avançados, os pesquisadores ponderam que mesmo que os trabalhos resultem em plantas mais resistentes, o melhoramento genético tem um limite. Se a temperatura subir mais do que 2°C com relação a sua faixa ideal, não terá como combater o problema pois a planta passa a ter dificuldade em fazer fotossíntese.

Em conjunto a essa busca pelas condições ótimas de cultivo para cada espécie, visando um cenário de plantio urbano residencial, que vem crescendo desde o primeiro trimestre após o início do isolamento social decorrente da covid-19, como pode ser visto a partir do crescimento pela busca do termo “kit de jardinagem” realizadas no Google (2021) (resultados da pesquisa relacionados com kits voltados para plantações em quintais), Figura 01, evidencia-se a necessidade de um maior foco em pesquisas relacionadas ao uso consciente da água utilizada para a irrigação, principalmente em regiões metropolitanas como a de São Paulo e Distrito Federal destacados em nota da Câmara dos Deputados (2018) como regiões que já enfrentam a falta de abastecimento de água em determinadas épocas do ano mesmo estando em áreas geográficas que não apresentavam essas dificuldades anteriormente, problema conhecido popularmente como crise hídrica, sendo uma questão social que assola o mundo inteiro, até mesmo o Brasil que abriga a maior reserva hídrica potável do mundo, em perspectiva mundial o World Resources Institute (WRI) (2015), expõe os dados de que no ano de 2040 mais de 3,5 bilhões de pessoas no mundo sofrerão com a escassez de água se medidas não forem tomadas, ocasionando assim riscos à segurança alimentar dos países, inclusive o Brasil, como já apontava o Centro de Pesquisas Meteorológicas e Climáticas Aplicada a Agricultura da Unicamp desde 2008, expondo assim uma extrema necessidade do uso consciente e correto desse recurso insubstituível para a vida de maneira geral. Principalmente em práticas agrícolas, visto que os

dados da Organização das Nações Unidas para Alimentação e Agricultura (2018) demonstram que em média 70% dos recursos hídricos disponíveis no mundo e extraídos para o consumo humano são destinados à agricultura.

Figura 01: Gráfico das pesquisas realizadas no Google com o termo “kit de jardinagem” entre 16/03/2019 e 16/03/2021.



Fonte: GOOGLE, 2021.

1.2 Objetivo

Este trabalho propõe-se a desenvolver uma arquitetura para um sistema de irrigação automático em um ambiente urbano residencial, aplicado ao cultivo em vasos de plantas. Juntamente com uma Interface Homem Máquina (IHM) para fins de supervisão e comandos remotos nas amostras. Interligando os periféricos de atuação e sensoriamento a partir do uso de um microcontrolador, o qual será responsável por condensar as informações e as disponibilizar ao sistema supervisório. De forma que um usuário administrador do sistema possa compreender o histórico das variáveis, e controlar o sistema a partir de um dispositivo com conexão à internet.

2. Referencial Teórico

Neste tópico serão embasadas algumas teorias e conceitos principais cujo conhecimento são necessários para uma melhor compreensão do desenvolvimento realizado.

2.1 Cultivo

De acordo com Marouelli (1996) em publicação para revista da EMBRAPA e transcrita para informativo no site oficial da instituição em 2006, para o cultivo de tomates (planta a qual será cultivada no desenvolvimento deste trabalho), da semeadura à emergência das plântulas, as irrigações devem ser leves e freqüentes, de modo a manter os primeiros 10 cm do solo sempre umedecidos. Nessa fase, o turno de rega deve ser de 1 a 2 dias, dependendo do tipo de solo e das condições climáticas. Em solos arenosos e/ou em regiões de temperatura elevada e de baixa umidade relativa do ar, o turno de rega deve ser diário. Irrigações frequentes também são recomendadas por ocasião do transplante. Neste caso, deve-se irrigar preferencialmente pela manhã, quando a temperatura é mais amena e as plantas estão geralmente túrgidas.

Dependendo do tipo de solo e do clima da região, as irrigações devem ser paralisadas 20 a 30 dias antes do início da colheita, quando as plantas apresentarem cerca de 20% de frutos maduros. Essa medida visa concentrar a maturação de frutos e aumentar a concentração de sólidos solúveis. Entretanto, em termos de produção de frutos, maiores produtividades podem ser obtidas irrigando-se até a ocasião em que cerca de 50% dos frutos estiverem maduros.

Dentre os vários critérios existentes para o manejo da irrigação, nenhum pode ser considerado padrão nem indicado para todas as situações. Métodos clássicos que permitem um controle bastante criterioso da irrigação, como o do balanço hídrico e da tensão de água no solo, baseiam-se no conhecimento das características físico-hídricas do solo, das necessidades específicas da cultura e de fatores climáticos relacionados à evapotranspiração. Dependem ainda do uso de equipamentos para o monitoramento da umidade do solo (tensiômetros, blocos de resistência elétrica, etc.) ou de equipamentos para a estimativa da evapotranspiração (tanque Classe A, termômetros, higrômetros, etc.). Essas informações e equipamentos, além de não estarem, em geral, ao alcance do irrigante, exigem conhecimentos técnicos específicos para seu manuseio.

De modo geral, as irrigações na região do Cerrado são feitas por aspersão, utilizando-se o pivô-central. No Vale do São Francisco, usa-se a irrigação por sulco, que consiste na distribuição

de água por meio de sulcos paralelos às fileiras de plantio. A água é geralmente conduzida por um canal principal, de onde é derivada para os sulcos, utilizando-se tubos plásticos denominados de sifões, com diâmetro de uma a duas polegadas. A distribuição da água pode ser feita também por tubos janelados, que possuem diversas aberturas reguláveis que permitem o controle da quantidade de água aplicada em cada sulco de irrigação.

O comprimento dos sulcos e sua declividade são determinados em função da textura do solo. Os sulcos devem ter 15 a 20 cm de profundidade e 25 a 30 cm de largura. As maiores dimensões são utilizadas para solos de baixa velocidade de infiltração.

Para a temperatura, a pesquisa de Geisenberg & Stewart (1986) chega nos resultados da tabela 1, que apresenta as temperaturas ideais para o cultivo de tomates.

Tabela 1: Temperaturas para o Cultivo de Tomates.

Estágio de desenvolvimento	Temperatura (°C)		
	Mínima	Ótima	Maxima
Germinação	11	16 a 29	34
Crescimento vegetativo	18	21 a 24	32
Pegamento de frutos (noite)	10	14 a 17	20
Pegamento de frutos (dia)	18	19 a 24	30
Desenvolvimento da cor vermelha	10	20 a 24	30
Desenvolvimento da cor amarela	10	21 a 32	40

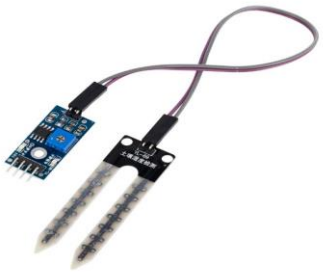
Fonte: GEISENBERG & STEWART, 1986.

2.2 Sensor de Umidade do Solo Higrômetro

Este sensor serve para detectar as variações de umidade no solo, sendo que quando o solo está seco a saída do sensor fica em estado alto, e quando úmido em estado baixo. O limite entre seco e úmido pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Contudo, para ter uma resolução melhor, é possível utilizar a saída analógica A0 e conectar a um conversor AD.

A saída analógica varia de 300 mV a 700 mV em solo úmido, e de 700 mV até 950 mV em solo encharcado ou água pura. Em solo seco ou pouco úmido varia de 0 (zero) a 300 mV.

Figura 02: Sensor de Umidade do Solo Higrômetro.

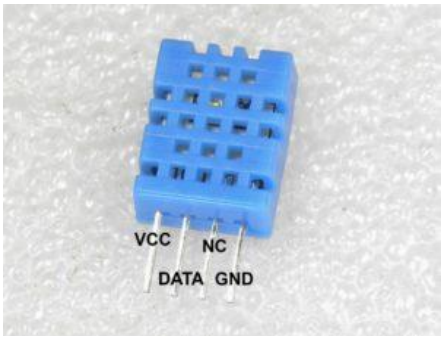


Fonte: THOMSEN, Adilson, 2016.

2.3 Sensor DHT11

O Sensor de Umidade e Temperatura DHT11 (2013) é um sensor de temperatura e umidade que permite fazer leituras de temperaturas do ar entre 0 (zero) e 50° Celsius e de umidade do ar entre 20 e 90%. O elemento sensor de temperatura é um termistor do tipo NTC (Negative Temperature Coeficient) e o sensor de umidade é do tipo HR202, o circuito interno faz a leitura dos sensores e se comunica a um microcontrolador através de um sinal serial de uma via.

Figura 03: Sensor de Umidade e Temperatura DHT11.



Fonte: MURTA, José, 2019.

2.4 Sensor de Fluxo de Água de 1/2"

Este Sensor de Fluxo de Água, modelo YF-S201, possui um rotor de água e um sensor de efeito Hall. Quando a água flui através do rotor, faz com que este gire, e as mudanças de velocidade

que variam de acordo com o fluxo de água são lidas pelo sensor de efeito hall, que emite o pulso de sinal correspondente. O sensor vem com 3 cabos conforme a tabela 2:

Tabela 2: Pinagem do Sensor de Fluxo de Água.

vermelho	alimentação de 5 a 24V
preto	terra
amarelo	saída do pulso do sensor de efeito hall

Contando os pulsos é possível calcular o fluxo de água. Note que este não é um sensor de precisão e que a taxa de pulsos pode variar um pouco dependendo do fluxo, da pressão e da orientação do sensor. No entanto é um excelente aliado para medições básicas.

O sinal do pulso é uma onda quadrada que se registra e se converte em litros por segundos com uma técnica de somar os valores enviados pelo sensor a cada segundo ao longo do tempo de calibração, e tirar uma média desses valores observados dentro do intervalo de acionamento. Ao injetar água na entrada do sensor e a armazenar em um recipiente de medida (copo de medidas em litros), teremos o valor em litros despejado no intervalo amostrado. Assim, descobrimos qual multiplicação devemos aplicar no valor lido pelo sensor para a resposta condizer com a água despejada no recipiente.

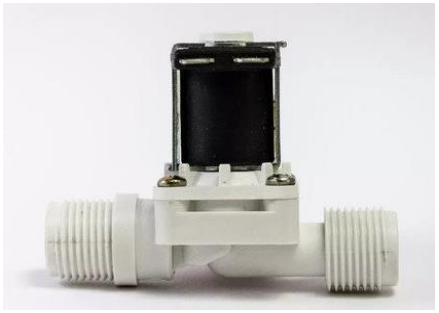
Figura 04: Sensor de Fluxo de Água de 1/2".

Fonte: MULTILOGICA-SHOP, 2021.

2.5 Válvula Solenóide de Entrada de Água

Esta Válvula Solenóide serve para controlar o fluxo de água do projeto. Quando alimentada em 220V ela permite a vazão de água, e corta o fluxo quando deixa de ser energizada.

Figura 05: Válvula Solenóide de fluxo de água 1/2" -220V.

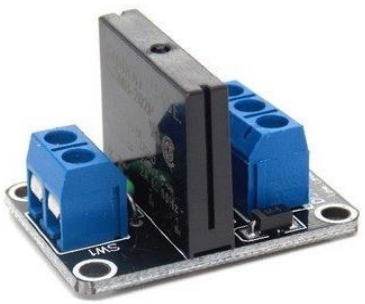


Fonte: VIDAL, Vitor, 2017.

2.6 Relé de Estado Sólido SSR 1 Canal

Com um módulo relé de estado sólido SSR de 1 canal você pode controlar equipamentos que normalmente são ligados em tomadas residenciais com apenas um pino do microcontrolador, o qual é responsável por fechar ou abrir a rede de energia onde o equipamento está conectado. O Relé de estado sólido (SSR - Solid State Relay, em inglês), se refere a um dispositivo semicondutor capaz de desempenhar as mesmas funções de um relé eletromecânico comum, com a diferença de não possuir partes ou contatos mecânicos. Isso aumenta a vida útil do relé, reduz a produção de ruídos durante o acionamento e aumenta a segurança de operação. O relé de estado sólido de 1 canal é ativado em estado baixo (LOW), com tensão entre 0 e 2,5V. Para desligar o relé, aplica-se de 3 a 5V no pino CH1.

Figura 06: Relé de Estado Sólido SSR de 1 Canal.



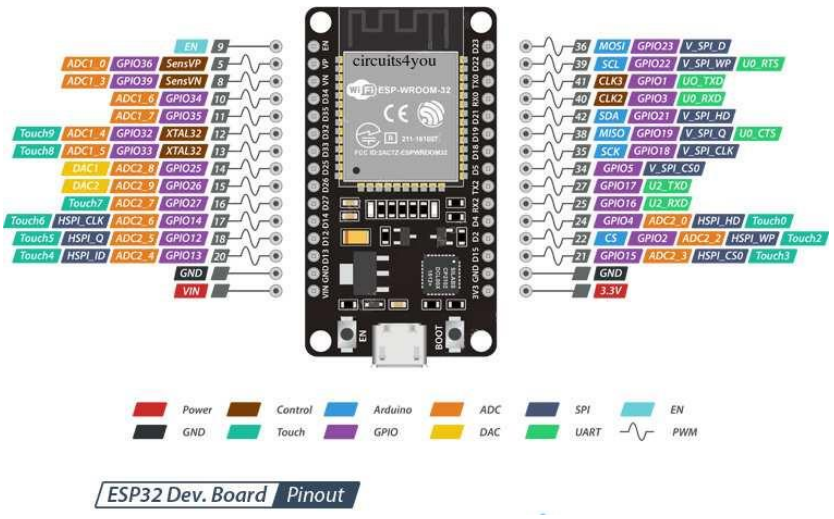
Fonte: ELETROGATE, 2021.

2.7 Microcontrolador ESP32

Conforme destacado por Bertoleti (2019) em seu livro Projetos com ESP32 e LoRa, ESP32 é o nome de um SoC (System on a Chip) (ou combo chip, conforme referenciado na documentação), o qual foi desenvolvido pela Espressiff System (2019). Ele dispõe de conectividade com Wifi e Bluetooth, e seu poder computacional dividido entre a CPU e as memórias. Possuindo um sistema Dual Core com duas CPUs Harvard Architecture Xtensa LX6, tendo dois núcleos, PRO_CPU para protocolo e APP_CPU para aplicação, no entanto, os objetivos desses não são fixos. Em seu chip ainda está incluso I/O 's, RTC (Real Time Clock), e suporte a comunicações diversas (SPI, I²C I²S, etc.). Além de suporte Low-Power e blocos de hardware dedicados à segurança. Com relação a sua CPU e memórias, são ROM de 448KB, SRAM de 520KB e duas memórias RTC de 8 KB. Sendo que sua memória externa suporta até quatro vezes o Flash de 16 MB. O espaço de endereço para o barramento de dados e instrução é de 4 GB e o espaço de endereço periférico é de 512 KB.

A Espressiff System tem uma placa de desenvolvimento com base no módulo ESP32 chamada ESP32-WROOM-32. Uma das placas de desenvolvimento criadas por outra empresa para demonstrar o ESP-WROOM-32 é o DevKit v1, feito pela DOIT. Ela é baseada também no microcontrolador ESP32 que possui suporte a Wifi, Bluetooth, Ethernet e baixa potência em seu único chip. Essa será usada no desenvolvimento deste trabalho.

Figura 07: Detalhamento da Pinagem do ESP32 DevKit v1.

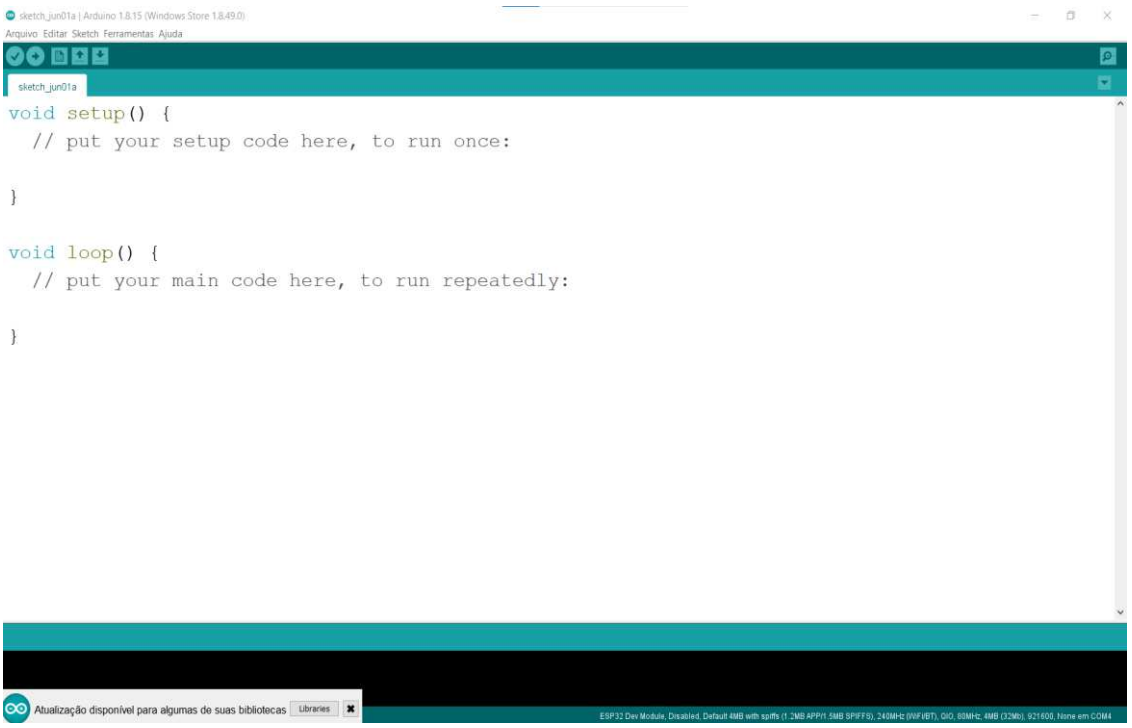


Fonte: OLIVEIRA, 2019.

2.8 Arduino IDE

Arduino (2018) é uma plataforma eletrônica de código aberto baseada em hardware e software projetados para serem fáceis de usar. Para realizar as configurações dos projetos, é utilizado o software open-source Arduino IDE, na escrita de códigos e no seu envio para dispositivos compatíveis, que podem ser tanto hardwares desenvolvidos pela própria Arduino, ou por outros desenvolvedores que possuem suporte na plataforma. Seu ambiente foi escrito em Java e Processing, sendo compatível com Windows, Mac OS X ou Linux. Para realizar as configurações, se utiliza da linguagem de programação Arduino (baseada em Wiring), consistindo em uma modificação da Linguagem C ++, sendo possível escrever códigos (sketchs) a partir dela. A Espressif Systems, fabricante de microcontroladores, disponibilizou o pacote para ser instalado no gerenciamento de placas da IDE, permitindo assim que a ESP32 possa ser programada através dessa IDE (Ambiente de Desenvolvimento Integrado) ao instalar as configurações da placa. Pela IDE, também é possível observar a troca de informações da comunicação serial, além de outras funcionalidades.

Figura 08: Arduino IDE.



2.9 Comunicação MQTT

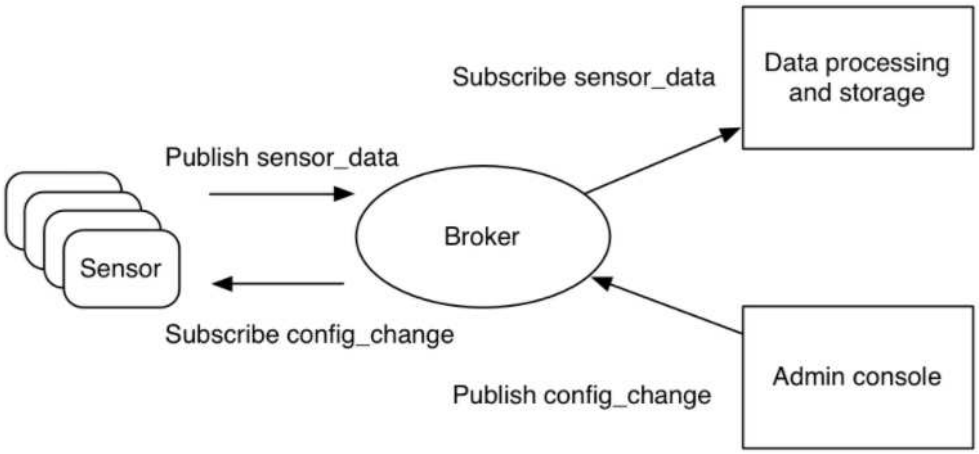
Para os dispositivos de Internet das Coisas (IoT), a conexão com a Internet é um requisito. A conexão com a Internet permite que os dispositivos trabalhem entre si e com serviços de backend. O protocolo de rede subjacente da Internet é o TCP/IP. Desenvolvido com base na pilha TCP/IP (Transmission Control Protocol/Internet Protocol), o MQTT (Message Queue Telemetry Transport) tornou-se o padrão para comunicações de IoT.

O MQTT foi inventado e desenvolvido inicialmente pela IBM (2017) (International Business Machines) no final dos anos 90. Sua aplicação original era vincular sensores em pipelines de petróleo a satélites. Como seu nome sugere, ele é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS (Organization for the Advancement of Structured Information Standards), com suporte nas linguagens de programação populares, usando diversas implementações de software livre.

O protocolo MQTT define dois tipos de entidades na rede: um message broker e inúmeros clientes. O broker é um servidor que recebe todas as mensagens dos clientes e, em seguida, roteia essas mensagens para os clientes de destino relevantes. Um cliente é qualquer coisa que possa interagir com o broker e receber mensagens. Um cliente pode ser um sensor de IoT em campo ou um aplicativo em um data center que processa dados de IoT.

Como as mensagens do MQTT são organizadas por tópicos, o desenvolvedor de aplicativos tem a flexibilidade de especificar que determinados clientes somente podem interagir com determinadas mensagens. O modelo de publicação e assinatura do MQTT para sensores de IoT é conforme a Figura 09.

Figura 09: Modelo de publicação e assinatura do MQTT para sensores de IoT.



Fonte: IBM, 2017.

Yassein (2017), na conferência internacional sobre engenharia (ICEMIS), caracteriza que o protocolo MQTT funciona principalmente como um canal para dados binários, e fornece flexibilidade nos padrões de comunicação. Sendo projetado para fornecer um protocolo de mensagens de publicação-assinatura. Neste padrão, quando um elemento da rede deseja receber uma determinada informação, ele a subscreve, fazendo uma requisição ao broker, o intermediário no processo de comunicação. Os elementos que desejam publicar informações, as enviam ao broker, que as reuni, e as repassa aos clientes que subscreveram os tópicos. Em publicação na 3ª Conferência Internacional sobre Software de Sistemas de Comunicação e Middleware, Hunkeler (2008) cita que em comparação com outras variantes de modelos que centralizam os dados, os sistemas de publicação/assinatura são comuns e amplamente difundidos na computação distribuída.

Conforme levantado pelo Professor Doutor Marcelo Barros no blog embarcados (2015), apesar do broker representar um elo de fragilidade na rede ao centralizar as comunicações, ele permite um desacoplamento entre as partes comunicantes, algo que não é possível em modelos de comunicação do tipo cliente/servidor, por exemplo. Vale lembrar que existem soluções de brokers redundantes ou operando em clusters, na tentativa de mitigar esta fragilidade.

As mensagens no MQTT são identificadas através de tópicos. O tópico lembra o conceito de URI (Uniform Resource Identifier), com níveis separados por barras (“/”). Clientes podem enviar diversos tópicos para o broker e os subscritores podem escolher os tópicos que desejam subscrever.

A conexão do cliente ao broker é feita via TCP para ambos os casos (subscritor ou publicador), sendo possível configurar login (usuário e senha) e uso de criptografia (SSL/TLS)(Secure Sockets Layer/Transport Layer Security). É possível encontrar também outros meios físicos com MQTT rodando, como em links seriais, por exemplo.

O processo de conexão estabelece um nível de qualidade de serviço (QoS, de Quality of Service), indicando como deve ser a relação entre os elementos comunicantes. São previstos três níveis de qualidade de serviço:

- QoS 0 (at most once): É o que conhecemos como "best effort", ou melhor esforço. Assemelha-se ao protocolo de transporte UDP (User Datagram Protocol), onde não se tem confirmação de entrega da mensagem. Quem envia também não tem a obrigação de manter a mensagem armazenada para futuras retransmissões;
- QoS 1 (at least once): Neste nível existe a confirmação de entrega de uma mensagem. Atende situações onde quem envia acaba gerando várias mensagens iguais possivelmente por um atraso na chegada de confirmação de recebimento. Neste caso, é garantido que uma delas terá o reconhecimento realizado. Note que existe um armazenamento da mensagem por parte de quem envia até a posterior confirmação;
- QoS 2 (exactly once): Garante que a mensagem seja entregue exatamente uma vez, com envio de confirmações de recebimento e confirmações de recebimento de confirmações de recebimento. Existem confirmações nos dois sentidos, para tudo que é trafegado. Enquanto uma mensagem não é confirmada, ela é mantida. É um caso mais próximo do protocolo de transporte TCP.

Vale ressaltar que não existe um QoS melhor ou pior, isto irá depender dos recursos disponíveis no seu sensor, de cada cenário de aplicação do MQTT, da qualidade do link de comunicação, etc. Cada nível de QoS é negociado entre o cliente e o broker, não entre o publicador e o subscritor. Assim, é possível ter uma publicação em QoS 0 e uma subscrição em QoS 2, para um mesmo tópico.

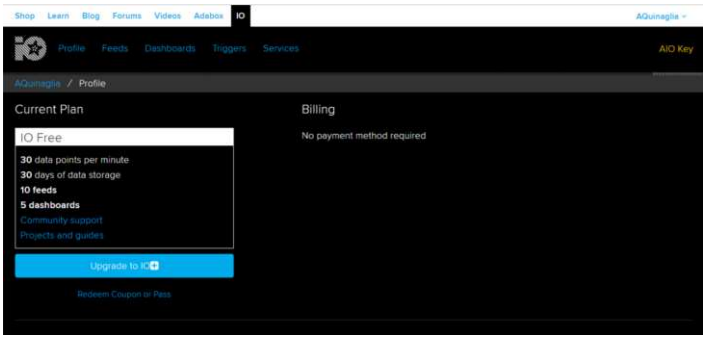
2.10 Adafruit IO

Adafruit IO é uma plataforma projetada pela empresa norte-americana Adafruit. Tem como objetivo exibir, responder e interagir com os dados de projetos diversos. Com a promessa de manter os dados privados (feeds de dados são privados por padrão) e seguros (nunca serão vendidos ou doados para outra empresa). Ela disponibiliza um armazenamento em nuvem e funcionalidades de

gerenciamento dos dados publicados a ela. É possível fazer projetos tanto em em Python como na linguagem Arduino (variação do C/C++), através do protocolo de comunicação MQTT. Se apresentando com uma opção de ferramenta para implementação de desenvolvimentos de IoT.

Para utilizar a plataforma é necessário criar uma conta, existindo dois tipos de planos de assinatura, um plano gratuito e um plano pago. Como é apresentado na tela inicial do Adafruit IO, (Figura 4). No plano gratuito, é possível realizar uma troca de até trinta dados por minuto, entre cliente e broker, caracterizando assim 1 envio a cada dois segundos. E esses dados são armazenados na nuvem por até 30 dias, e excluídos do banco de dados após este período. Nessa troca de dados entre os pontos, são permitidos 10 Feeds (equivalentes aos tópicos da comunicação MQTT), e a partir destes dados 5 Dashboards podem ser desenvolvidas. Já para o plano pago, a taxa de comunicação dobra, passando para sessenta dados por minuto, e os Feeds e as Dashboards são ilimitadas.

Figura 10: Tela Inicial Adafruit IO.



Fonte: TREENCE, Todd, 2015.

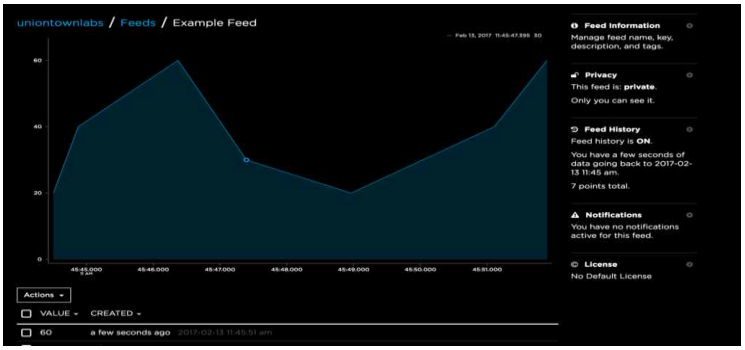
Dentro da plataforma existe um esquema de login e senha, que permite que cada projeto possua uma conta distinta para recepcionar seus dados com segurança, ambas as informações podem ser encontradas no botão *AIO KEY* presente no canto superior direito da Figura 4. Quando um cliente envia informações a plataforma Adafruit IO, o usuário pode interagir com esses dados a partir dos Feeds e das Dashboards, que serão melhor explicadas a seguir.

2.10.1 Feeds

Conforme apontado por Todd Treece (2015) no treinamento disponibilizado no portal Adafruit, os feeds são o núcleo do sistema Adafruit IO. O feed contém metadados sobre os dados enviados ao Adafruit IO. Isso inclui configurações para definir se os dados são públicos ou privados, em que licença os dados do sensor armazenados se enquadram e uma descrição geral dos

dados. O feed também contém os valores dos dados do sensor que são enviados ao Adafruit IO a partir do dispositivo cliente, é preciso criar um feed para cada fonte exclusiva de dados que enviar ao sistema. Por exemplo, em um projeto com três sensores de temperatura e seis sensores de umidade, precisará criar nove feeds, um feed para cada sensor.

Figura 11: Exemplo de feed da plataforma Adafruit IO.



Fonte: TREENCE, Todd, 2015.

2.10.2 Dashboards

Todd Treece (2015), em outro treinamento presente no portal Adafruit, caracteriza as Dashboards como ferramentas que permitem visualizar dados e controlar projetos conectados ao Adafruit IO, a partir de qualquer navegador web (computadores, smartphones, etc). Para desenvolvê-las, faz-se o uso de Widgets, que são interfaces gráficas que facilitam a interação do usuário com a aplicação. Que tem como exemplo: gráficos, controles deslizantes, botões, displays e outros recursos disponíveis que permitem colocar o projeto em funcionamento sem a necessidade de qualquer código personalizado. As Dashboards permitem que o usuário visualize e manipule os Feeds conforme seu desejo, e é possível relacionar até cinco dados diferentes em cada gráfico.

Figura 12: Exemplo de Dashboard da plataforma Adafruit IO.

Fonte: TREENCE, Todd, 2020.

3. Metodologia

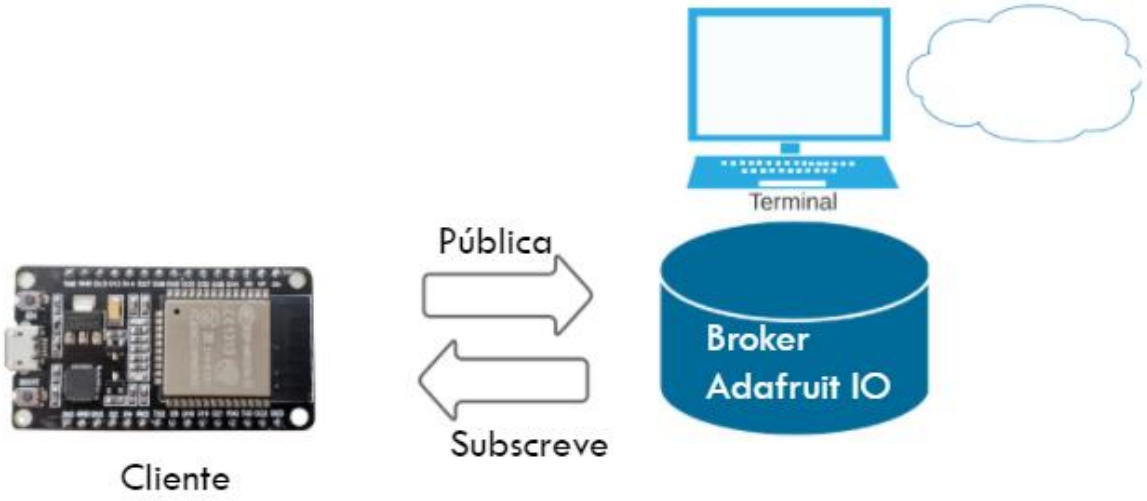
Durante a elaboração do sistema automático de irrigação e controle de temperatura para estufas, o desenvolvimento foi dividido em etapas nas quais serão descritas a seguir.

3.1 Arquitetura

Neste projeto, é proposta uma arquitetura que tem como característica a possibilidade de interligar um sistema de irrigação a uma central de dados, que abrigue dados distintos, de forma que dentro de um universo crescente de interconexão de objetos via internet, este arranjo tenha a adaptabilidade de se conectar em uma rede maior, a qual lida com outros tipos de informações. De forma que, em modelos de casas inteligentes, para aquelas que tiverem o espaço para vasos de plantas, esse sistema apresentado possa facilmente ser incorporado ao restante da casa, ao enviar os dados à central, que os processa e sustenta a automação residencial. Esse tipo de arquitetura funciona para os diversos tipos de aplicações, como pode ser observado pelo exemplo do trabalho de conclusão de curso do Augusto Quinaglia (2019), apresentado ao Instituto de Ciência de Tecnologia de Sorocaba, da (UNESP), onde ele demonstra um arranjo que monitora as variáveis ambientais de um estoque de peças aeronáuticas, evidenciando assim, que a partir deste modelo de arquitetura, onde há um nó cliente e um servidor do tipo broker MQTT, é possível condensar os mais diversos tipos de informações dentro desse arranjo.

Durante o desenvolvimento elaborado, para fins de validação da arquitetura do sistema de irrigação automático, visando que além dessa característica de escalabilidade, o sistema cumpra os requisitos que permitam a um usuário supervisionar as condições sensoriadas, e a atuação ocorra de maneira automática quando os limites estabelecidos são atingidos, além de permitir também um controle via comandos remotos nas variáveis de atuação; Foi feita a montagem de três vasos de 10 litros contendo mudas de tomate, onde três sensores de temperatura e de umidade do ar avaliam as condições das amostras, em conjunto com outros três sensores de umidade do solo. Os quais enviam os dados a um microcontrolador ESP32, que também recebe os valores do sensor de fluxo de água. Esse sistema controla uma válvula solenóide e um ventilador usando relés de estado sólido. E o conjunto está conectado à internet por meio de uma conexão wifi estabelecida com um roteador, o que permite o sistema enviar e receber informações de um broker MQTT, o qual pode ser acessado pelo portal Adafruit IO. Para a montagem do Nó e para sua execução, são necessários cabos conectores para interligar os periféricos, e também um acesso a um ponto de água para ser a fonte na irrigação. Interligando o sistema conforme a Figura 13, que simplifica as informações que serão apresentadas com mais detalhes nas Figuras 14 e 16, sendo o cliente correspondente ao nó sensor/atuador.

Figura 13: Arquitetura desenvolvida.

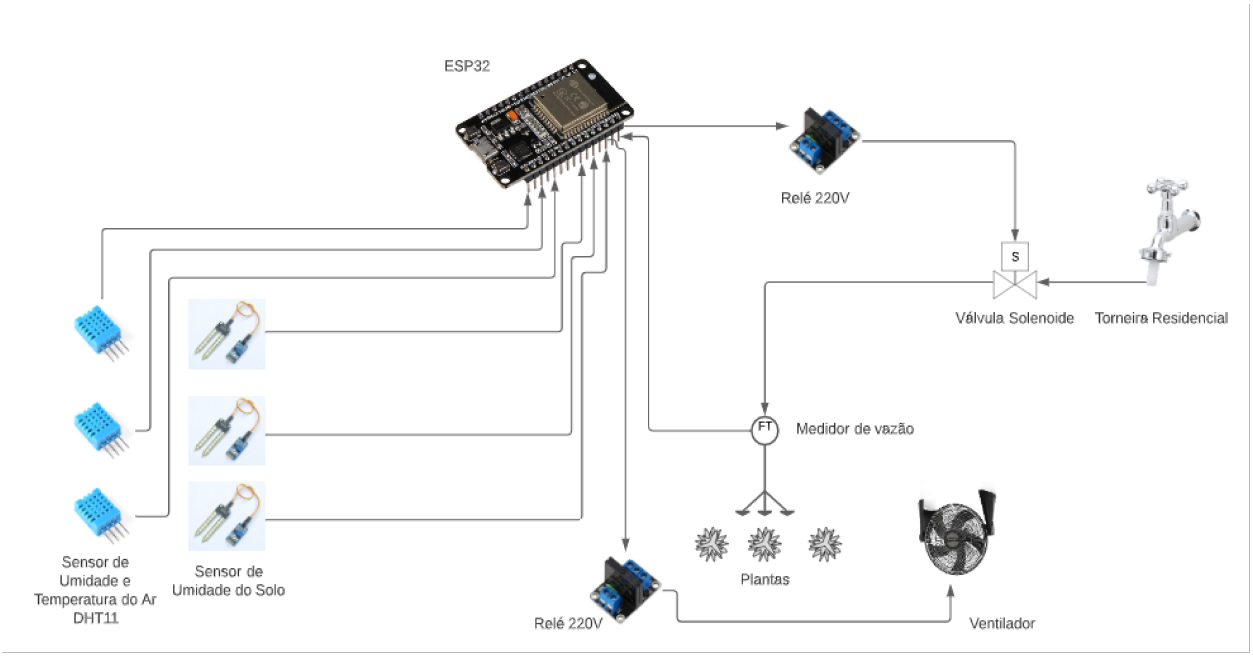


3.2 Implementação do Nó Sensor e Atuador

Para demonstrar a proposta de arquitetura, foi montado um nó sensor/atuador usando uma placa de desenvolvimento DOIT Esp32 DevKit v1 da empresa DOIT, que faz uso do módulo ESP-WROOM-32 que é baseado no microcontrolador ESP32 e que possui suporte a Wifi, Bluetooth e Ethernet em seu chip.

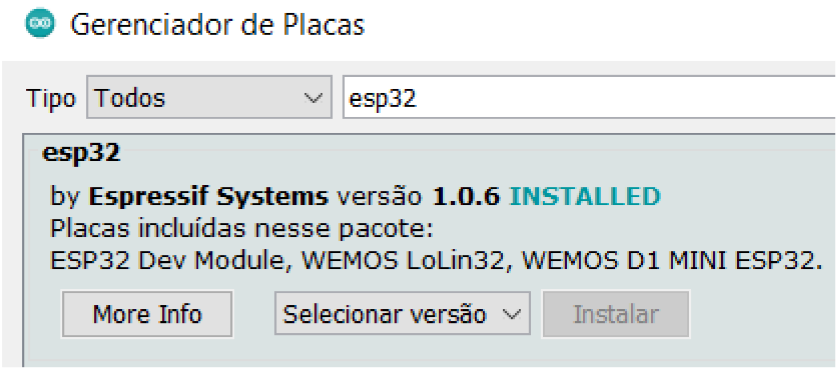
Esta placa foi conectada a sensores DHT11 e também a sensores de umidade do solo relacionados aos três vasos de amostra montados para a validar a arquitetura do sistema proposto, além de receber informações do medidor de fluxo de água e a partir de um tratamento das condições observadas, o módulo ESP32 comanda dois relés que acionam uma válvula solenóide e um ventilador.

Figura 14: Arranjo do Nó Sensor/Atuador.



Para desenvolver as lógicas de leitura e atuação do Nó foi usada a IDE arduino para programar o módulo ESP32, para isso foi necessário primeiro instalar um conversor USB-Serial de forma que fosse possível comunicar o computador e o chip. Isso foi feito com o uso do driver "CP210x USB to UART Bridge Virtual COM Port (VCP)" correspondente ao Sistema Operacional. Também é preciso instalar a placa ESP32 na IDE do Arduino, para isso a Espressif Systems, fabricante de microcontroladores, disponibilizou o pacote para ser instalado no gerenciamento de placas da IDE, que é possível ser encontrado ao adicionar o link para o JSON (JavaScript Object Notation) da Espressif em URL's adicionais presente no item preferências da aba arquivos da IDE.

Figura 15: Instalação da Placa ESP32.



A partir daí, a placa ficou em condições de receber o programa usando a plataforma Arduino IDE. Código este que irá realizar o sensoriamento e a atuação nas amostras e também configurar a comunicação com um broker MQTT.

Para avaliar a umidade e a temperatura ambiente em que os três vasos com as amostras estão condicionados, foi utilizado um sensor DHT11 em cada um, este que possui uma biblioteca que ajuda a interpretar seus dados. Ele é um equipamento que possui quatro pinos de conexão, onde apenas três deles foram usados, sendo VCC, GND e Data. O sensor de umidade é capacitivo e o sensor de temperatura é um termistor NTC, isto é, um resistor sensível à variações de temperatura. Dentro do sensor existe um microcontrolador que faz as medições e transmite os valores no formato digital através de um pino de saída. Para ler os dados utiliza-se as funções da biblioteca DHT: ".readHumidity();" e ".readTemperature();". Que já lhe devolve os valores de umidade relativa em uma escala de 20 a 90% e o de temperatura dentro de uma faixa de 0 a 50 °C.

Já para o sensoriamento da umidade da terra no interior dos vasos foi usado um sensor do tipo Higrômetro que consiste em duas partes: uma sonda que entra em contato com o solo, e um pequeno módulo contendo um chip comparador LM393, que vai ler os dados que vêm do sensor e enviá-los para o microcontrolador (ESP32) . Como saída, ele tem um pino D0 (digital output), que fica em nível 0 (zero) ou 1 dependendo da umidade, e um pino de saída analógica (A0) (analog output), que possibilita monitorar a umidade do solo com maior precisão usando uma porta analógica do microcontrolador.

A medição da quantidade de água injetada para os vasos das plantas a cada irrigação é feita por um sensor de fluxo de água de modelo YF-S201, este sensor usa os princípios do eletromagnetismo, de forma que, quando os líquidos passam pelo sensor, a ação do fluxo atinge as aletas de uma turbina no sensor, fazendo com que a roda gire. O eixo da roda da turbina é conectado a um sensor de efeito Hall, de modo que a cada giro é gerado um pulso e, monitorando esse pulso pelo microcontrolador, o sensor pode ser usado para determinar o volume de fluido que passa por ele.

Para isso, é preciso habilitar um pino do microcontrolador para trabalhar como interrupção de descida, de forma que é contada a quantidade de bordas de descida da onda quadrada do pulso, o que a partir de uma constante de calibração e de médias, é possível determinar quantos mililitros de água são injetados nos vasos a cada atuação de irrigação. O que será melhor elucidado no tópico 3.4 desta seção.

Uma válvula solenóide é usada para abrir a passagem de água, quando se deseja controlar a

irrigação das plantas. Ao ser alimentada em 220V ela permite a vazão de água, e corta o fluxo quando deixa de ser energizada. Essa válvula é acionada por um relé de estado sólido, que é ativado em estado baixo (low), com tensão entre 0 (zero) e 2,5V. Para desligar o relé se aplica de 3 a 5V no pino CH1. Este que conecta e desconecta a rede de energia em 220 Volts que alimenta a válvula solenóide. O relé de estado sólido é um dispositivo semicondutor capaz de desempenhar as mesmas funções de um relé eletromecânico comum, com a diferença de não possuir partes ou contatos mecânicos.

Utilizando também um outro relé de estado sólido, um ventilador 220 V é ligado e desligado de acordo com a temperatura naquele momento. Usando como base os dados de temperatura obtidos a partir de uma média da leitura dos três sensores DHT11, foi configurado que para valores acima de 24 ºC o ventilador é ligado e abaixo de 21 °C é desligado. O setpoint foi escolhido com base nos dados da pesquisa de Geisenberg & Stewart (1986) que chegaram nos resultados da tabela 1, com relação às temperaturas ideais para o cultivo de tomates.

Completando assim o arranjo da Arquitetura do nó sensor e atuador, o qual terá o seu código fonte, descrito no tópico 3.4 desta seção.

3.3 Implementação da Comunicação MQTT

Entende-se que o Nó sensor e atuador apresentado anteriormente já consegue realizar as atividades propostas com relação aos cuidados das plantas cultivadas em vasos, porém, para informar as condições controladas a um usuário administrador do sistema precisamos optar por pelo menos uma de duas possíveis soluções; a primeira é desenvolver um display local, e a segunda é enviar os dados para algum outro dispositivo que possa reproduzi-los, e assim, a partir de uma das abordagens tornar possível supervisionar as condições controladas.

Pensando na qualidade da interface e dos recursos disponíveis para a análise do sistema, nota-se que desenvolver uma estação de supervisão local usando algum tipo de display, iria exaurir os diversos recursos do microcontrolador que detém as informações a serem expostas, desde o uso dos terminais do barramento para conectar e controlar a interface, até uso de memória Flash e RAM.

Por entender que dentre os diversos dispositivos e placas como alguns modelos de Raspberry Pi, ou de Beaglebones, que são equipamentos que possuem recursos para sustentar os periféricos do Nó em sua leitura e atuação nas condições do ambiente, e que também conseguem controlar uma Interface Homem Máquina (IHM) em paralelo a isso, todos apresentam um preço para aquisição superior a um microcontrolador mais simples que tenha acesso a algum tipo de conexão que o

permita realizar comunicação. De forma que placas como o DOIT Esp32 DevKit v1 se apresentam como uma opção viável por ter capacidade de sustentar a arquitetura do nó, como outros microcontroladores também podem, mas em conjunto a isso ela já possui integrada ao seu chip uma antena de rede que a permite realizar conexão Wifi. De forma que a partir desta comunicação ela pode enviar as informações do Nó para outro dispositivo.

Mas, para compararmos o valor de aquisição do microcontrolador entre um que possui recursos para sustentar uma estação de supervisão local, e outro que não possui, é preciso considerar que ao enviar os dados vamos aproveitar dispositivos pré-adquiridos pelo usuário, como seu smartfone ou computadores e notebooks da residência onde a horta se encontra, para serem os equipamentos que recebem os dados vindo do Nó, e a partir deles apresente de forma organizada e precisa as condições monitoradas pelo arranjo ao usuário que irá administrar o sistema.

Então, de frente a esta condição, onde é preciso enviar os dados adquiridos pelos sensores do Nó a outros dispositivos via Wifi, precisamos escolher um protocolo de comunicação adequado para as características do sistema proposto. Onde possa garantir a entrega dos dados de maneira precisa e segura, sem que as informações do sistema fiquem visíveis para terceiros, e de forma que os dados continuem privados.

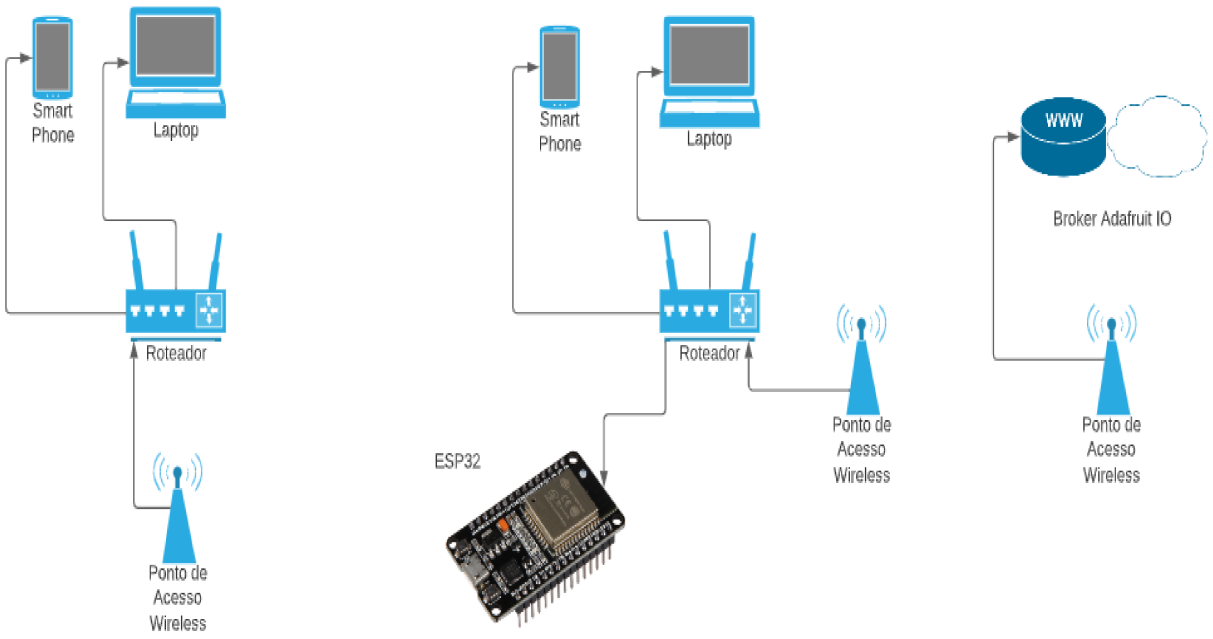
Considerando este cenário apresentado, podemos sondar alguns possíveis protocolos como o HTTP (Hyper Text Transfer Protocol), desde seu uso em uma implementação de uma página web, até no uso de REST (Representational State Transfer) que não é exatamente um protocolo mas sim um padrão bem estabelecido para transmissão de dados, no qual todo computador moderno e linguagem de programação tem suporte para REST API (application program interface). Porém, pensando no crescente universo da automação residencial e no aumento de dispositivos com suporte para integrarem uma rede de Internet das Coisas, podemos desenvolver a comunicação do Nó sensor via protocolo MQTT (Message Queue Telemetry Transport) , no qual o possibilita uma fácil integração a uma possível central de dados, ou seja, o broker. E como nosso sistema é proposto a um ambiente urbano, este cenário nos permite que em uma ampliação da estrutura controlada seja possível integrar dados da horta com informações de outros Nós, seja equivalentes ao desenvolvido neste trabalho, ou outros com objetivos de sensoriamento e atuação distintos da irrigação, como uma geladeira conectada ao broker por exemplo, ou um sistema de vigilância de câmeras. Por isso, ao se demonstrar como um protocolo que permite escalar e ampliar a arquitetura de comunicação em diversas direções, o MQTT foi o escolhido para o uso neste trabalho.

Dentro do universo de opções para implementar a comunicação MQTT, diversos tipos de arquitetura são possíveis, desde a estruturação de um broker em meu domínio usando como exemplo

uma base com o Mosquitto fornecido pela Eclipse, ou o uso de um servidor online como é o caso do broker disponibilizado pela empresa Adafruit, intitulado de Adafruit IO. Para este trabalho, foi escolhido o broker Adafruit por dois motivos, o primeiro foi por este ser uma ferramenta que não demanda uma máquina para executá-lo, ou seja, a própria fornecedora já hospeda a aplicação em seus servidores. O outro motivo são as opções de interface gráficas disponíveis a partir de Dashboards, e também em detalhamento dos Feeds, sem contar a capacidade de armazenamento dos dados mais recentes. Sendo esses, exemplos de recursos que permite que para seu desenvolvimento não seja necessário desenvolver um seguindo cliente para subscrever os tópicos publicados pelo Nó sensor, que teria o objetivo de ser uma estação de supervisão, visto que o próprio Adafruit IO nos fornece as condições de desenvolver esta interface pelo uso dos widgets das Dashboards.

A arquitetura da comunicação escolhida para este desenvolvimento se apresenta conforme a Figura 16, onde é possível que o usuário administrador do sistema observe as condições e os históricos das amostras a partir de uma conexão com a internet, seja interno ou externo a rede Wifi em que o Nó sensor está conectado. Precisando apenas acessar o portal Adafruit IO em um navegador e fazer o login em uma conta criada previamente, seja usando um smartphone, computador, notebook, tablet ou algum outro dispositivo com navegação na internet similar aos citados anteriormente. Onde o usuário terá acesso a dashboards que possuem detalhes a respeito das amostras sensoriadas e controladas pelo Nó. Além de poder armazenar os dados trocados com uma data de até um mês anterior ao momento em questão, e também baixar estes dados em uma planilha no formato “.csv” sempre que desejar. Dentro desta arquitetura, o Nó sensor é um cliente que publicará alguns tópicos ao broker Adafruit, e a partir dele, uma interface de supervisão será desenvolvida para análise das condições.

Figura 16: Arquitetura de Comunicação e Supervisão dos dados.



Para desenvolver esta arquitetura é necessário que duas frentes de ações sejam consideradas, a primeira delas é criar uma conta para receber os dados dentro do arranjo fornecido pela Adafruit, e com ela obter algumas informações específicas como seu nome de usuário e a sua senha, que são possíveis visualizar dentro do portal Adafruit IO, além de configurar as dashboards que serão a interface principal entre o administrador e os dados. A segunda frente consiste em configurar o Nó sensor para publicar os tópicos desejados e também subscrever aquelas informações que deseja receber do broker.

Para criar a conta, deve-se acessar o endereço: "io.adafruit.com" e lá clicar em "*Get Started for Free*" e seguir com o seu cadastro. Nessa etapa você cria um Username, no qual é o mesmo que deverá ser referenciado na programação do Nó como o usuário destino no qual os dados serão enviados ou recebidos dentro do arranjo do broker Adafruit. Para descobrir a senha que deve ser inserida, também durante a conexão do Nó, deve-se clicar em "my key" presente na tela inicial do portal Adafruit, de forma que sua senha lhe é informada. A partir daí, você pode tanto criar os Feeds (equivalentes aos tópicos que serão publicados e subscritos) pela página Web ou diretamente pelo Nó, ou seja, caso um cliente envie um valor em um Feed ainda não existente, esse é criado automaticamente.

Para este trabalho, os seguintes Feeds foram estabelecidos em uma etapa inicial para realizar testes quanto aos modelos de irrigação:

Tabela 3: Tópicos da Comunicação em uma etapa inicial do desenvolvimento.

	Umidade do Solo	Água Injetada	Temperatura do Ar	Umidade do Ar	Solenóide	Ventilador
Vaso 1	/feeds/umidadesolo.vaso1	/feeds/litros1	/feeds/temperatura	/feeds/umidade	/feeds/solenoide	/feeds/ventilador
Vaso 2	/feeds/umidadesolo.vaso2	/feeds/litros2				
Vaso 3	/feeds/umidadesolo.vaso3	/feeds/litros3				

Já em uma etapa final do desenvolvimento, após validar os modelos de irrigação e definir um padrão para todos os vasos, de forma que eles serão irrigados ao mesmo tempo, temos os seguintes feeds:

Tabela 4: Tópicos da Comunicação em uma etapa final do desenvolvimento.

	Umidade do Solo	Água Injetada	Temperatura do Ar	Umidade do Ar	Solenóide	Ventilador
Vaso 1	/feeds/umidadesolo.vaso1	/feeds/litros	/feeds/temperatura	/feeds/umidade	/feeds/solenoide	/feeds/ventilador
Vaso 2	/feeds/umidadesolo.vaso2					
Vaso 3	/feeds/umidadesolo.vaso3					

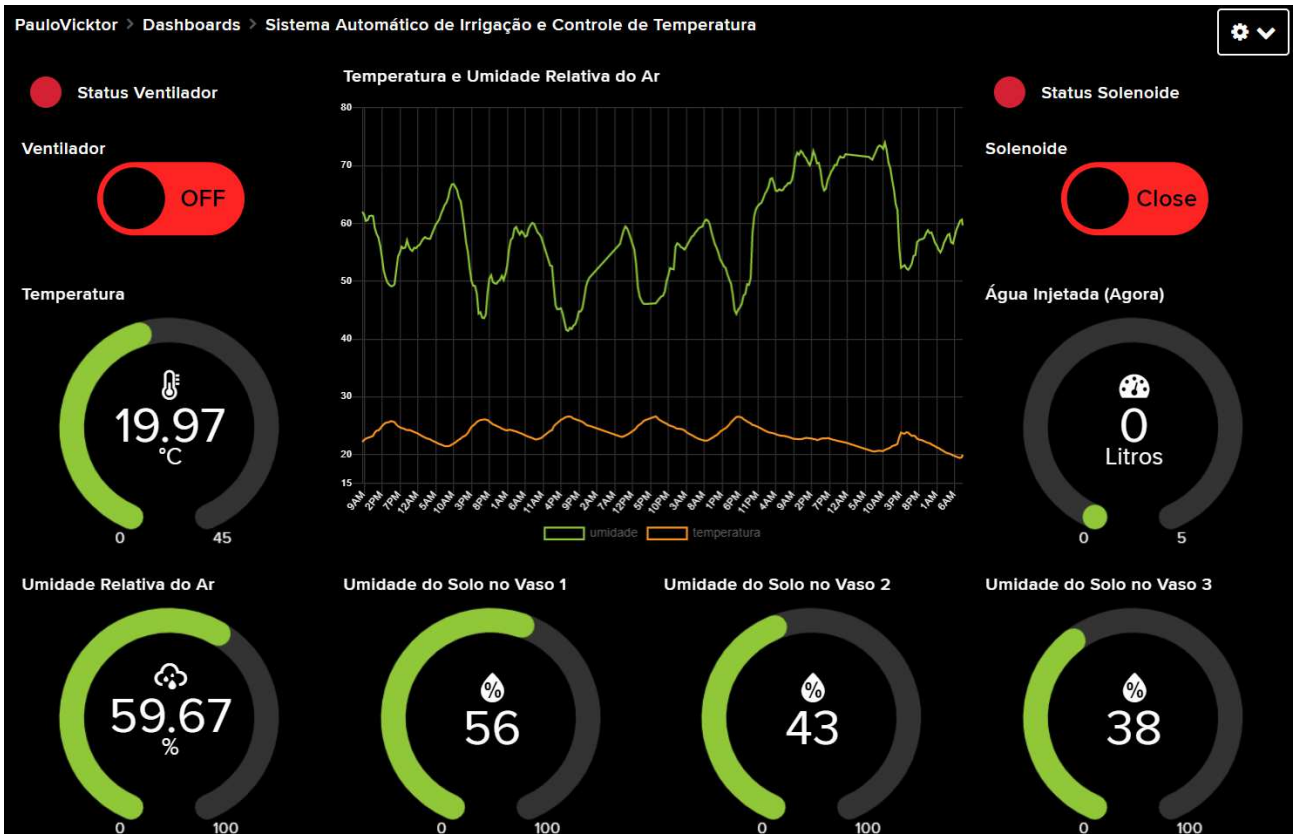
Com um tópico configurado e estabelecido, podemos observá-los na seção Feeds do Portal Adafruit IO. E clicando neles temos acesso a uma tabela que nos informa a data da publicação do dado e seu respectivo valor.

Pelo lado do Nó sensor, o cliente da comunicação, deve-se configurar os tópicos a serem publicados, como também o valor e a qualidade do serviço escolhida para cada um dos envios do tópico. Esta parte será melhor descrita na parte 3.4 desta seção.

A partir dos Feeds recebidos pelo broker após o envio dos dados por um cliente, podemos desenvolver dashboards para melhor interpretar estas informações. Elas permitem que você visualize dados e controle os projetos conectados ao Adafruit IO, a partir de qualquer navegador da web moderno. Widgets como gráficos, controles deslizantes e botões estão disponíveis para ajudar a colocar o projeto em operação, sem a necessidade de um código personalizado. Uma dashboard

foi desenvolvida para este trabalho, visando supervisionar as condições das plantas sensoriadas e realizar comandos manuais nos relés de atuação, um para a válvula solenóide de irrigação e o outro para o ventilador. Conforme a figura 17.

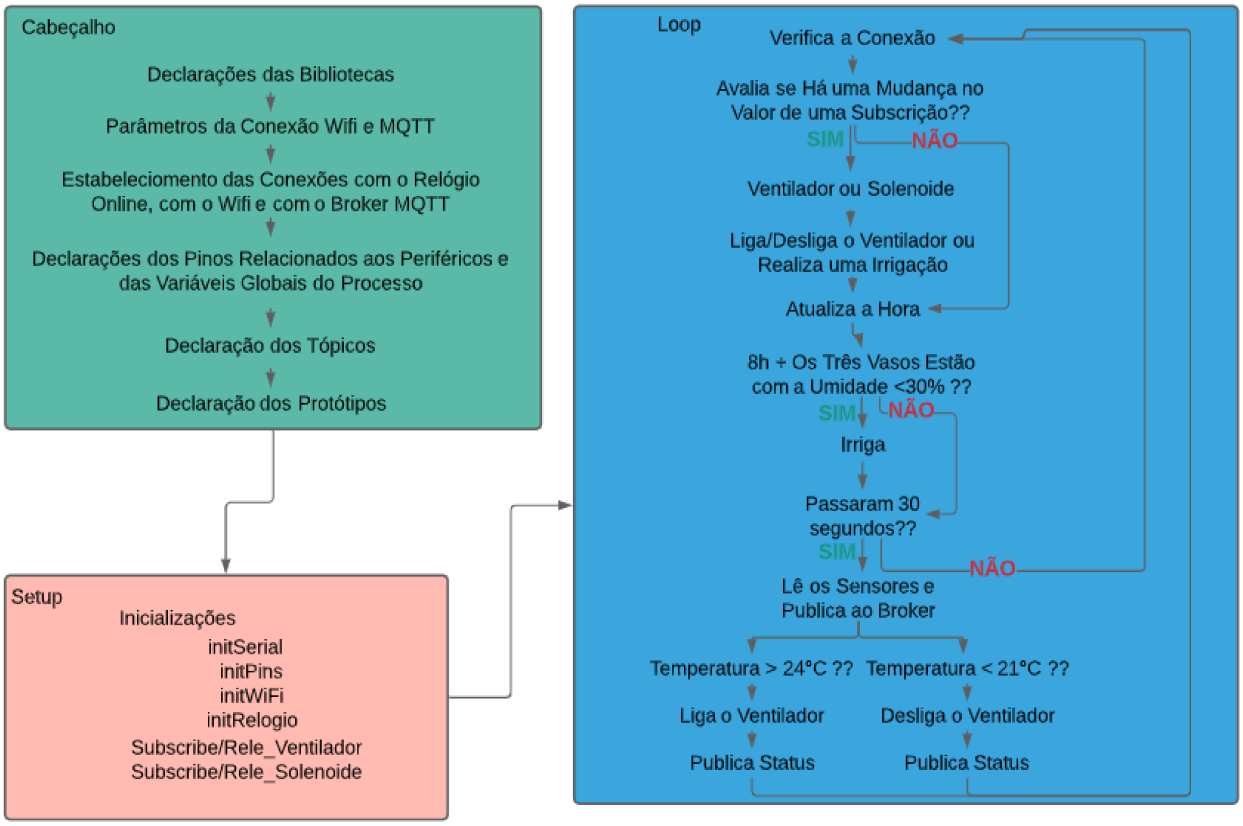
Figura 17: Dashboard de Supervisão do Sistema.



3.4 Aquisição dos Dados

Para a execução deste trabalho foi necessário programar o microcontrolador ESP32 e os componentes da placa de desenvolvimento DOIT Esp32 DevKit v1, com o uso do software Arduino IDE. O código elaborado para cumprir com as propostas será descrito em partes ao longo desta seção, e está disponível em sua íntegra no Anexo 1. Seguindo o fluxo da Figura 18:

Figura 18: Fluxo do código fonte do nó.



3.4.1 Cabeçalho e Declarações

Foram incluídas cinco bibliotecas para dar suporte ao desenvolvimento da programação, sendo uma correspondente ao sensor DHT11, outra para estabelecer conexão Wifi, uma terceira para obter os valores de um relógio online, uma quarta para configurações da comunicação MQTT, e uma última que estabelece configurações extras com as características de um cliente MQTT. Sendo: DHT.h; WiFi.h; NTPClient.h; Adafruit_MQTT.h; e Adafruit_MQTT_Client.h.

Após isso, são declaradas o usuário e a senha da conexão Wifi, na qual o microcontrolador irá conectar, em seguida foram inseridas as configurações da comunicação MQTT, apontando o servidor no qual iremos conectar, a porta de rede, além do usuário e da senha da respectiva conta Adafruit IO.

Também é feita a declaração dos pinos da placa DOIT Esp32 DevKit v1 com os correspondentes nomes que os representará ao longo do código, de forma a propiciar um melhor entendimento nos momentos de leituras e escritas dos periféricos associados aos respectivos pinos.

Em sequência, foram declaradas as variáveis globais usadas no processo, aquelas que são usadas por mais de uma função (prototype).

Já para estabelecer os tópicos que serão publicados e subscritos do broker Adafruit IO, foram declarados 10 Feeds inicialmente, e após concluir os testes e chegarmos ao momento da tabela 04, os valores de litros 1, 2 e 3 foram substituídos por um único tópico, "/feeds/litros", visto que todos os vasos passaram a ser irrigados ao mesmo tempo, mudando assim para 8 Feeds. Sendo que destes, 2 são tanto publicados quanto subscritos, por serem relacionados ao Status da válvula de irrigação e do ventilador. As quais precisam publicar ao broker quando houver uma atuação automática, e subscrever os dados quando um comando remoto for enviado.

Para realizar o desenvolvimento do algoritmo, foram criadas funções que condensam e organizam as diferentes situações do código, as quais serão explicadas individualmente a seguir, e seguindo a sequência da exposição, seus protótipos estão declarados logo abaixo das configurações dos Feeds.

3.4.2 Execução do Código

Após a declaração das funções, o código parte para a execução em si do algoritmo. Primeiro com a seção de Setup e depois com a de Loop.

No setup, são chamados os protótipos de inicialização do algoritmo, além de anunciar a subscrição nos tópicos dos relés. Já no loop, primeiro é verificado a conexão com o broker MQTT, caso não esteja conectado ele chama a função que restabelecerá a comunicação. Na sequência, são avaliadas as subscrições do Nó cliente, que corresponde às duas variáveis de atuação (solenóide e ventilador), chamando a função específica para isso. Na sequência é atualizada a hora atual.

No momento da Tabela 03, foram realizados três tipos de irrigações diferentes, sendo duas definidas por horários e uma por um setpoint de umidade do solo. Essa parte aparece comentada no código exposto no Anexo I. Já no momento da Tabela 04 foi aplicado apenas um modelo para todos os vasos, que avalia a umidade do solo dos três vasos com relação a um setpoint uma vez ao dia, e com relação a resposta, atua.

Além disso, a cada trinta segundos é feita uma leitura em todos os sensores, verifica-se a qualidade dos valores obtidos, e os publicam ao broker MQTT, e por fim checa a necessidade de ligar ou desligar o ventilador observando a temperatura atual.

3.4.3 Detalhamento dos Prototypes

3.4.3.1 Inicializações

Temos quatro prototypes de inicializações, um para o monitor serial, outro para setar os pinos como entrada ou saída e suas condições iniciais, um terceiro para estabelecer a conexão Wifi e um último que conecta com o relógio online, são eles: initSerial(); initPins(); initWiFi(); e initRelogio().

3.4.3.2 Avaliação da Subscrições

Quando essa função é chamada dentro do Loop (*"avalia_subscri();"*), ouvimos as mensagens recebidas. Caso houver uma mudança em alguma, é checado qual dos dois status foi alterado e a partir de sua resposta é aberto ou fechado o circuito da rede de energia. No caso do comando para abrir a válvula solenóide, é passada a chamada de atuação de irrigação, na qual será melhor detalhada mais à frente, e após isso retorna o status de close para o tópico, de forma que não permite que alguém abra o fluxo de água indiscriminadamente.

3.4.3.3 Irrigação

Nesta função *("atua(Adafruit_MQTT_Publish titulo);")* primeiro se abre a válvula solenóide acionando o relé, e envia o status de aberto ao broker, em seguida, avalia-se quantas vezes a interrupção de descida do pino do sensor de fluxo de água é realizada, a cada vez que é chamada essa interrupção é incrementado o valor de uma variável em 1, essa leitura é feita 3 vezes em um intervalo de 1 em 1 segundo, e retirada uma média ao final. Após os três segundos em que a válvula fica aberta e são feitas as leituras, é passado o comando para fechá-la novamente, em seguida o valor injetado nas plantas no momento é publicado para o broker, e também o status de fechado para a válvula.

Para calibrar o valor lido pelas bordas de descida da onda quadrada do sensor, e torná-lo correspondente a quantidade de água injetada em litros, foi colocado um recipiente medidor de volumes na saída da mangueira que a válvula solenóide abre e fecha o fluxo de água, após a atuação, se comparou o valor em litros depositados no recipiente com o valor lido pelo sensor. Ao comparar os valores, pode-se aplicar uma constante multiplicadora que correlacione ambos, para definir o valor mais próximo das condições de pressão da rede de água na qual a válvula solenóide e o medidor de fluxo estão instalados, esse teste foi realizado diversas vezes. Mas, é importante pontuar que a rede de água pública não mantém exatamente a mesma pressão ao longo do dia em sua linha, assim, caso a pressão na qual a água chega até a torneira alimentadora da válvula, e por consequência do medidor de fluxo, não seja a mesma de quando a calibração foi realizada, o valor lido sofre divergências com relação ao real volume usado na irrigação. Os maiores problemas se apresentam em momentos onde há ar presente na tubulação, visto que os intervalos de irrigação são muito curtos (por volta de 3 a 5 segundos) de forma que um pequeno volume de ar que movimenta a turbina do sensor já apresenta uma grande diferença entre a quantidade de água real injetada e o valor colhido pelo equipamento referente a volume de fluidos (água + ar) que passou pela tubulação.

3.4.3.4 Publicação ao Broker

Temos duas chamadas para publicação, uma para valores Float *(“publica(Adafruit_MQTT_Publish titulo, float valor);”)* e outra para Strings *(“publica_status(Adafruit_MQTT_Publish titulo, String valor);”)* essa segunda que converte as Strings em Char, para o correto envio ao broker. Quando uma dessas funções é chamada, é enviado ao broker o valor correspondente ao tópico que foi passado em seu cabeçalho.

3.4.3.5 Leitura dos Sensores

Quando essa função é chamada, são feitas as leituras dos três sensores de umidade do solo, utilizando de uma leitura analógica para cada um, os quais recebem um valor entre 4095 e 0, de forma que 4095 representa um solo seco e 0 um solo encharcado, e a partir deste valor convertemos essa escala para ficar entre 0 e 100, onde 0 é para um solo seco e 100 úmido. Também são realizadas as leituras de temperatura e umidade relativas do ar dos sensores DHT11 a partir da biblioteca do equipamento, para estes, se soma os valores dos três sensores para obter a média, a qual é enviada ao broker em outra parte do algoritmo.

3.4.3.6 Conexão com o broker

Para se conectar com o broker é passado o comando “mqtt.connect()”, a partir de seu resultado é avaliado se a conexão ocorreu com sucesso ou se houve alguma falha, e no caso de falha, tenta-se conectar novamente após cinco segundos.

3.5 Modelos de Irrigação

Com o objetivo de compreender as reações das amostras a partir de diferentes tipos de atuações, foram propostos três modelos de irrigação distintos. Sendo um primeiro configurado para injetar água no período da manhã, um segundo no período noturno, e um terceiro permitindo a irrigação apenas em situações em que o valor da umidade do solo era inferior a 30% na escala do sensor. Para realizar essa configuração, foi preciso de um auxílio constante do administrador do sistema, pois havia apenas uma mangueira de irrigação e foi preciso redirecionar ela para o vaso correto durante esse período de análise e setup. Dessa forma, foi necessário que cinco minutos antes das irrigações do vaso 1 e do vaso 2 o administrador direcionasse a mangueira para o vaso correto, e após o fluxo de água ser despejado, devolver a mangueira para o vaso 3 que aguardava a umidade de sua terra secar até o valor de 30% da escala de medição do sensor.

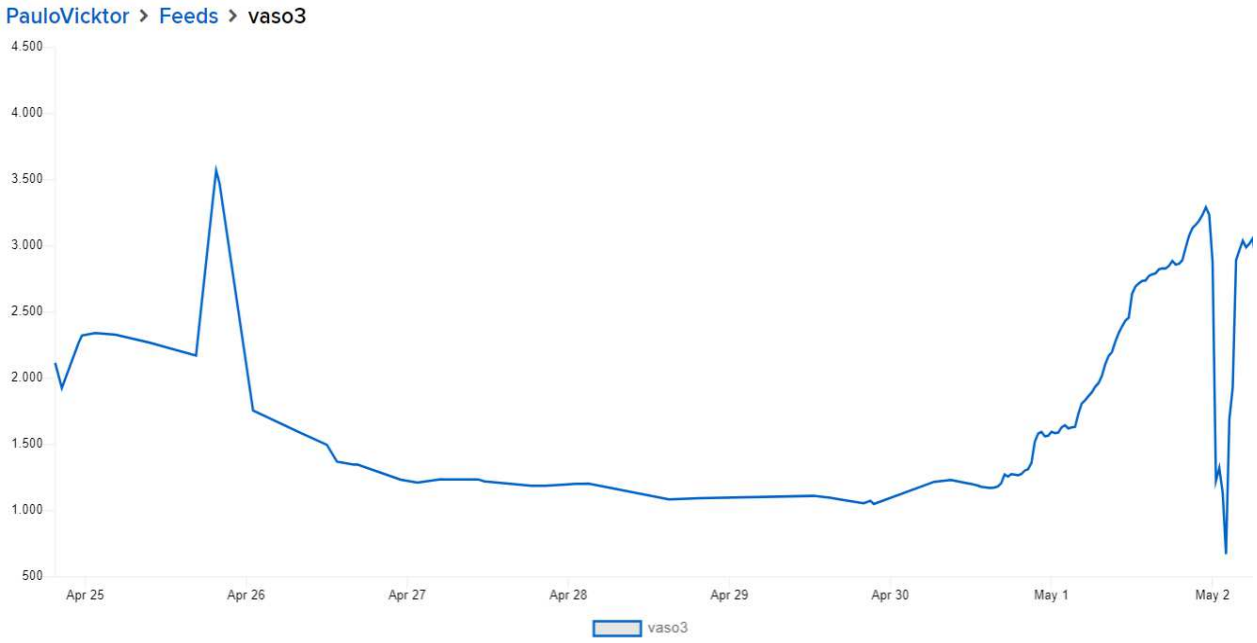
Nesse ponto do desenvolvimento, era medido individualmente o volume de água injetada no vaso a cada irrigação, possuindo um tópico para cada vaso, como pode ser observado na Tabela 03. Para o vaso 1, foi configurado que a irrigação acontece no período noturno. A ideia deste modelo buscava entender se a curva da umidade do solo demonstraria uma decaimento mais rápido ou mais devagar em comparação a curva relacionada ao vaso 2, que foi programado para ser irrigado no período da manhã. Já para o vaso 3, um setpoint relacionado a umidade do solo foi estabelecido, de forma que a irrigação era feita apenas quando o setpoint fosse atingido. Após esses testes, o sistema foi desligado e reconfigurado para executar um único modelo de irrigação para os três vasos, de forma que durante a abertura da válvula, a mangueira pudesse despejar água em todos os vasos, para isso a ponta da mangueira foi obstruída e furos foram abertos na lateral da mangueira em alguns pontos, amarrando um pano em volta deles para diminuir a pressão e o esguicho da água, estes pontos foram colocados no interior dos vasos, de forma que ao abrir a válvula solenóide a água é despejada em todos os vasos a partir destas aberturas. Nessa configuração final, a irrigação só ocorre durante a parte da manhã (às 08 horas) em dias em que a umidade da terra dos três vasos está abaixo de 30% da escala de medição do sensor. E, ao ser realizada, é medido o volume de água enviada para os três vasos ao mesmo tempo, diferente do modelo de testes que os avaliava individualmente.

4. Resultados e Discussões

4.1 Resultados dos Modelos de Irrigação

Ao implementar os modelos de irrigação feitos para testes, que tiveram o intuito de avaliar o melhor comportamento do algoritmo de atuação automática, logo nos primeiros dias ficou explícito que a frequência de irrigação de forma diária, indiscriminadamente do valor da umidade do solo, resultava em plantas com excesso de água em seus vasos. Como pode ser observado por um recorte do gráfico da umidade da terra presente no vaso 3, visto na Figura 18, que representa os valores brutos (antes da conversão para porcentagem) lidos pelo sensor de umidade do tipo higrômetro, de forma que, solos encharcados tem um valor inferior ao de uma terra seca:

Figura 19: Umidade do Solo do Vaso 3 em Valores Brutos do sensor (0-4095) ao longo de Oito Dias.

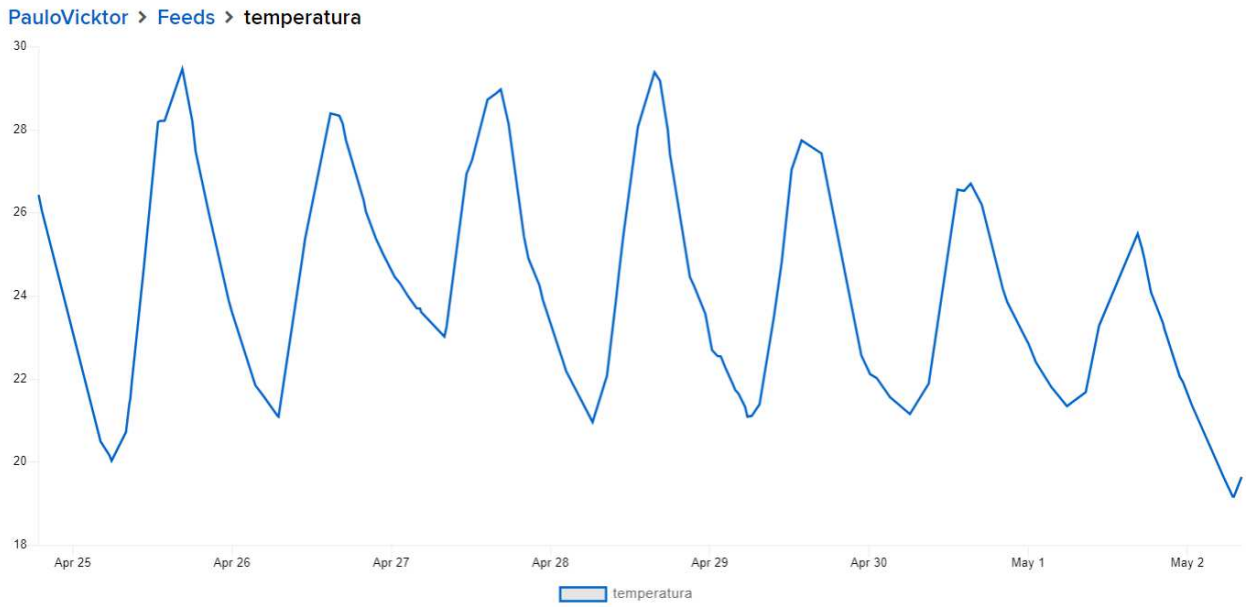


Como pode se observar pela curva, no dia 26 de abril, após um período em que a terra secou rapidamente devido a incidência do sol sobre a superfície do recipiente, e como foi um dia com poucas nuvens no céu tornando a irradiação mais expressiva, a umidade do solo reduziu de maneira visível. E assim, foi realizada a atuação de irrigação, porém, do dia 26 para o dia 27 iniciou-se uma sequência de dias nublados nos horários que há incidência direta do sol nas amostras, o que ocasionou os vasos permanecerem em situação de sombra em grande parte do dia, e como isso, a umidade do solo não apresentou grandes variações entre os dias 27 e 30 de abril. Isso ocorreu mesmo que a temperatura nestes dias não estivesse discrepante com relação aos outros dias, como pode ser visto na Figura 19, demonstrando assim que estar nublado ou não, pode influenciar mais a umidade

da terra do que a temperatura do ar naquele dia em si.

No dia 2 de Maio, após a umidade da terra no vaso 3 voltar a diminuir, foi realizada uma irrigação quando o sensor observou uma condição pré configurada de setpoint. Nesse dia o céu estava aberto e com poucas nuvens, a atuação ocorreu por volta do início da manhã, e após a incidência direta do sol, por volta do meio do dia, quando a irradiação acumulou no plano, houve uma redução rápida da umidade, mesmo sendo um dia mais frio com relação aos dias anteriores. Evidenciando ainda mais, o efeito que o sombreamento dos dias nublados causam nas condições monitoradas dos vasos.

Figura 20: Temperatura do Ar em Graus Celsius ao longo de Oito Dias.



Essa situação mostrou de forma clara que qualquer ato de irrigação deve ser feito apenas após uma avaliação da umidade da terra do vaso naquele instante. Pois, como é visto na relação entre as Figuras 18 e 19, mesmo em dias com temperaturas similares, são necessários outros fatores como a incidência do sol para resultar em efeitos no ambiente monitorado.

Agora, pensando com relação à análise esperada para a comparação entre os modelos de irrigação dos vasos 1 e 2, onde se correlaciona o impacto de atuar logo no início do dia, em comparação com regar os vasos à noite. Seus resultados foram impactados pelo ponto apresentado na análise do vaso 3, pois, assim como entre os dias 27 e 30 de abril, uma irrigação que seja realizada indiscriminadamente do valor da umidade da terra no instante pode resultar em uma terra encharcada, o que também é prejudicial para uma planta. Porém, observando o ocorrido para o vaso 03 demonstrado na Figura 18, podemos compreender uma correlação de um dia com pouco sol com o período noturno. Pois sem a incidência do sol, não há grandes variações na umidade do solo, visto

que no cultivo feito em vasos, se não for por evaporação, apenas a absorção das plantas interfere na umidade da terra.

Com isso, sabendo que para o modelo final de irrigação será avaliado o valor da umidade do solo, de cada um dos vasos uma vez ao dia, com base em um horário ideal para essa avaliação. E, considerando o que foi percebido ao longo da etapa de testes, este trabalho opta pelo horário do período da manhã para essa avaliação das umidades dos solos, visando condicionar água o suficiente para as plantas absorverem da terra durante um dia ensolarado, desta forma, caso o valor para todos os vasos estejam abaixo de um setpoint pré estabelecido em 30%, às 8 horas da manhã, ocorrerá a atuação.

4.2 Resultados da Interface de Supervisão

A dashboard desenvolvida para a supervisão do sistema de irrigação cumpriu bem o seu papel. Por este trabalho utilizar da plataforma Adafruit IO, como a base para a comunicação do nó sensor, foi possível montar em poucos passos uma tela, apresentada na figura 17, que reúne os valores da última alteração de cada tópico publicado para o broker. Por essa tela, também pode-se visualizar o histórico dos Feeds reunidos em um gráfico. E é permitido realizar ações manuais via comandos remotos nos relés de controle da válvula solenóide e do ventilador. De forma que, atua-se nestes periféricos a partir da dashboard de supervisão presente no Portal Adafruit IO. Porém, não é permitido abrir a válvula de água por um tempo indefinido, pois, ao enviar um comando, uma ação de irrigação será executada, e ao seu fim a válvula é fechada novamente. Pois dentro do universo dos vasos amostrados, não há sentido em permitir que a válvula solenóide seja aberta indiscriminadamente. Já para o ventilador, caso envie um comando para o desligar ou ligar ao seu relé, ele permanecerá neste estado até que alguma condição do controle automático identifique a necessidade de atuar nele para manter a performance ideal.

4.3 Resultado da Arquitetura Proposta

Com relação ao objetivo proposto de desenvolver um sistema que controle automaticamente as ações de irrigação de vasos de plantas presentes em ambientes residenciais, e que disponibilize os dados avaliados em uma plataforma online, acessada via um navegador web. A arquitetura apresentada cumpre com as expectativas. Sendo que, o Nó que sustenta a aplicação utiliza 50% do espaço de armazenamento para programas do microcontrolador, alocando 12% do disponível para memória dinâmica, e deixando 287980 bytes para variáveis locais.

Durante seu desenvolvimento, o sensor de fluxo de água do modelo YF-S201 apresentou resultados insatisfatórios, pois, seu sensoriamento que é feito ao gerar uma onda quadrada que pode ser utilizada para contar quantas bordas de descida houveram dentro de um intervalo de tempo, para determinar quantas voltas o eixo da turbina do sensor realizou, e assim estipular um volume de água transportado, não apresenta precisão em sua obtenção de dados, pois condições como uma mudança na pressão da água vinda da torneira, influenciam de forma evidente na aquisição e calibração do equipamento. Visto que a velocidade com que a água passa pelo sensor muda conforme a pressão da rede, e o número de voltas realizadas pela turbina não correspondem ao mesmo volume de água injetado que foi considerado durante o momento de ajuste e configuração de sua leitura.

Dessa forma, esta baixa precisão na leitura do sensor de fluxo nos força a buscar um outro tipo de equipamento para quantificar o uso da água no sistema, com foco em equipamentos que nos garanta que o valor registrado como o volume de água transportado é correspondente ao volume que fluiu pela via, independente da pressão momentânea e que não perca a precisão durante uma alteração na velocidade em que o fluxo foi transportado pela tubulação. O que nos permitiria somar o volume de água usado para a irrigação ao longo das semanas, o que não foi realizado durante e o desenvolvimento deste trabalho pois a constante necessidade de calibração do sensor gerou um erro no valor registrado.

Mas, com relação aos outros periféricos do nó (relés, DHT11, e sensor Higrômetro) o conjunto como um todo cumpriu com o que foi proposto, desde a aquisição dos dados pelo Nó, passando pelo envio dos dados, até a supervisão feita no portal Adafruit. De forma que, a partir do sistema, foi possível uma maior compreensão de como as variáveis do ambiente se comportam em relação a determinadas condições climáticas, visto a possibilidade de avaliar as amostras a partir de um acesso ao portal Adafruit IO. Além de realizar atuações na válvula solenóide e no ventilador tanto por comandos remotos quanto por ação automática.

4.4 Resultado da Comunicação

No desenvolvimento realizado para validar a arquitetura proposta, a troca de informações entre o nó e o broker foi explorada acima da necessidade real solicitada pelas características do sistema, visto que publicar o valores relacionados a temperatura e a umidade, em uma frequência de duas vezes por segundo, é excessivo, visto que foi configurado a qualidade de serviço para o tipo um, a qual garante a entrega de cada publicação de um tópico no mínimo uma vez. Este excesso é caracterizado pois a variação destas condições ocorrem em uma taxa mais lenta, o que permitiria que fosse configurados intervalos maiores entre os envios de dados ao broker, se fosse redefinido

para 5 minutos entre uma publicação e outra, seria reduzido o uso da rede em 10 vezes considerado com o que foi usado neste trabalho. Sendo essa uma característica que reforça o resultado positivo da arquitetura.

Pensando no cenário levantado, em que este arranjo de irrigação se conecte com uma central de automação residencial, é expressamente recomendado que seja aplicado este aumento do intervalo entre cada publicação, pois outros nós concorrerão pelo uso da largura de banda disponível na rede, e um uso exagerado pelo nó sensor e atuador de irrigação poderia gerar uma sobrecarga deste, sem real necessidade de demandar dessa frequência de comunicação.

E, considerando uma situação onde mais tópicos necessitem ser publicados ou subscritos, e essa abordagem de enviar todos ao broker sequencialmente, e depois esperar um intervalo para realizar as publicações novamente, cause gargalo-los na rede durante os momentos de publicação. É possível mitigar essa sobrecarga separando os dados em pacotes de publicações, os quais seriam reunidos com relação a sua criticidade para o correto funcionamento do sistema, e enviados em filas, distribuindo a troca de informações dentro de intervalos, de forma que todos não fossem enviados em uma única execução do loop.

4.4 Discussões

Por meio das constatações que puderam ser feitas neste trabalho de conclusão de curso, é preciso pontuar que ao buscar um sistema que auxilie os administradores durante o cultivo de vegetais, é importante controlar também a iluminação do ambiente, ação que não foi abordada durante esse trabalho, e que deve ser considerada em desenvolvimentos futuros. Pois, a iluminação é um fator chave no crescimento de uma planta, e sem um sistema atuador que complemente a luz natural do sol, a produtividade fica comprometida em dias nublados ou chuvosos, e as plantas se tornam mais suscetíveis às estações do ano, visto que em cada uma o dia possui um tamanho diferente com relação a noite, o que influencia nos ciclos de luz das plantas e em suas possibilidades de crescimento. O que poderia ser contornado utilizando um sistema suplementar de iluminação para compensar as necessidades dos vegetais.

Outro ponto levantado é a possibilidade de tubulações e válvulas distintas para a irrigação de cada vaso, de forma que ao configurar o setpoint da umidade do solo para permitir que ocorra a irrigação automática, cada vaso poderia ser irrigado individualmente, o que garantiria uma precisão maior no controle da umidade. Porém, pensando no cenário de uma residência, se não fosse desenvolvido um esquema extra de tubulações, que tivesse uma entrada e três saídas, seria necessário utilizar de três pontos similares a torneiras, o que não é comum ter tantas disponíveis em

uma só casa, o demandaria uma melhor avaliação da implementação, ou não, desta melhoria.

Assim, pensando neste cenário em que as três amostras são irrigadas simultaneamente, caso uma tenha necessidade de ser irrigada, e a outra ainda possua uma porcentagem considerável de água, o sistema tem que optar por deixar uma abaixo do setpoint ou levar a outra ao excesso.

5. Conclusão

A partir das considerações apresentadas ao longo deste trabalho, é possível concluir que a arquitetura proposta para o sistema de irrigação automático e controle de temperatura, para vasos de plantas no ambiente urbano, cumpre com o que foi proposto. Pois, desde sua coleta de dados, até a correta disponibilização das informações ao utilizar do protocolo MQTT para enviar as informações ao broker, apresentaram respostas condizentes e satisfatórias. Pois, esse sistema desenvolvido permitiu uma melhor compreensão dos efeitos que os diversos tipos de ações do ambiente causam nas condições de cultivo de vegetais. E, a partir dessas compreensões, um modelo final de irrigação foi proposto e implementado, o qual irriga as plantas uma vez ao dia, com base no valor atual da umidade da terra nos vasos, evitando excesso ou falta de água no cultivo.

O acesso a estação de supervisão, se mostrou efetiva no papel de transmitir noções do histórico do ambiente amostrado aos usuários administradores do sistema de irrigação, pois a partir dela, além de todas as informações das amostras que são disponibilizadas, também é possível realizar comandos remotos, contemplando assim as situações onde o administrador deseja realiza uma atuação que não foi pré configurada no controle automático.

E, interpretando os valores disponibilizados na interface, foi possível compreender que nos cultivos em vasos, a irradiação do sol diretamente em sua superfície pode ocasionar uma redução expressiva na umidade do solo presente ali, e em contrapartida, em dias nublados a umidade sofre poucas variações. O que indica que uma irrigação precisa não pode estar associada apenas a um intervalo de tempo, mas também a umidade naquele instante. Pois, caso todos os dias seja destinado um fluxo de água aos vasos, pode ocorrer o afogamento das plantas. Fato esse que direcionou o modelo final desenvolvido a considerar o valor da umidade do solo das três amostras antes de realizar uma atuação, apenas permitindo a abertura do fluxo de água caso elas estejam abaixo de um setpoint pré determinado.

Dessa forma, este modelo de irrigação para o cultivo de vegetais demonstra um uso mínimo de água para sua manutenção. Visto que apenas quando é constatada a real necessidade de repor a umidade ao vaso é permitido que o sistema realize uma ação de irrigação.

Já com relação aos recursos utilizados durante o desenvolvimento de validação da arquitetura, foi possível concluir que o nó implementado com o uso do microcontrolador ESP32, apresentou um desempenho aceitável. Visto que cumpriu com seus objetivos de sensoriamentos e atuações, concomitantemente manteve a comunicação com o broker Adafruit IO, mesmo ao

ser aplicada uma taxa de transferência de dados entre os pontos excessiva com relação ao real demandado pelo sistema.

A comunicação com base no protocolo MQTT também apresentou resultados positivos, pois permitiu que o arranjo de irrigação controlado pelo nó, representado neste trabalho pelo microcontrolador ES32, esteja conectado a qualquer dispositivo com acesso à internet. O que torna possível que os administradores da horta, ou da estufa residencial, tenham acesso às condições históricas e de tempo real de suas plantas, independente de onde estejam. Característica que permite uma correta manutenção das condições dos vegetais durante seu desenvolvimento, sem necessitar de uma pessoa presencialmente para verificar a situação do ambiente e as necessidades demandadas pelas amostras.

Bibliografia

ARDUINO. **What is Arduino?**, 2018. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 12 jun. 2021.

AGEITEC. **Temperatura no Cultivo de Soja**, 2011. Disponível em: <https://www.agencia.cnptia.embrapa.br/gestor/soja/arvore/CONT000fzr67cri02wx5ok0cpoo6aeh331my.html>. Acesso em: 06 jun. 2021.

EMBRAPA. **Cultivo de tomate para industrialização**, 2003. Disponível em: <https://sistemasdeproducao.cnptia.embrapa.br/FontesHTML/Tomate/TomateIndustrial/irrigacao.htm>. Acesso em: 25 jul. 2021.

EMBRAPA. **Soja em números (safra 2019/20)**, 2020. Disponível em: <https://www.embrapa.br/soja/cultivos/soja1/dados-economicos>. Acesso em: 06 jun. 2021.

BARROS, Marcelo. **MQTT – Protocolos para IoT.** 2015. Disponível em: <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: 06 jun. 2021.

BERTOLETI, Pedro. **Projetos com ESP32 e LoRa** . Editor do NCB, 2019.

DE LUCENA, Leandro Pessoa; DA SILVA, Claudio Eurico Seibert Fernandes. **Modelos de agricultura urbana para a segurança alimentar: um estudo comparativo entre Singapura e Brasil**. Revista Ibero-Americana de Ciências Ambientais, v. 9, n. 3, p. 379-397, 2018.

ELETROGATE. **Relé de Estado Sólido SSR 1 Canal.** 2021. Disponível em: <https://www.eletrogate.com/rele-de-estado-solido-ssr-1-canal>. Acesso em: 06 jun. 2021.

Espressif Systems. **ESP32 Series**, 2021. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 06 jun. 2021.

FAO**. FAO e CNA lançam estudo sobre agricultura irrigada brasileira**, 2018. Disponível em: <http://www.fao.org/brasil/noticias/detail-events/pt/c/1107498/>. Acesso em: 06 jun. 2021.

GEISENBERG, C.; STEWART, K. Field crop management. In: **The tomato crop**. Springer, Dordrecht, 1986. p. 511-557.

GOOGLE. **Google Trends**, 2021. Disponível em: <https://trends.google.com.br/trends/explore?date=2019-03-16%202021-03-16&geo=BR&q=kit%20jardinagem>. Acesso em: 06 jun. 2021.

GOVERNO, Federal. **Crise Hídrica**, 2018. Disponível em: <https://www2.camara.leg.br/atividade-legislativa/estudos-e-notas-tecnicas/publicacoes-da-consultoria-legislativa/fiquePorDentro/temas/crise-hidrica-mar-2018>. >. Acesso em: 06 jun. 2021.

GOVERNO, Federal. **Agricultura Familiar**. 2020. Disponível em: <https://www.gov.br/agricultura/pt-br/assuntos/agricultura-familiar/agricultura-familiar-1>. Acesso em: 06 jun. 2021.

HUNKELER, Urs; TRUONG, Hong Linh; STANFORD-CLARK, Andy. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In: **2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)**. IEEE, 2008. p. 791-798.

LIU, Thomas. **Digital-output relative humidity & temperature sensor/module DHT22 (DHT22 also named as AM2302)**. Aosong Electronics.[En línea]. Disponible: https://www. sparkfun. com/datasheets/Sensors/Temperature/DHT22. pdf [Último acceso: 09/2018], 2013.

MAROUELLI, Waldir Aparecido; E SILVA, WL de C.; DA SILVA, H. R. **Manejo da irrigação em hortaliças**. Brasília^ eDF DF: EMBRAPA-SPI, 1996.

MULTILOGICA-SHOP. **Sensor de Fluxo de Água YF-S201.** 2021. Disponível em: <https://multilogica-shop.com/sensor-de-fluxo-de-%C3%A1gua-yf-s201>. Acesso em: 06 jun. 2021.

MURTA, José. **Sensores DHT11 e DHT22: Guia básico dos sensores de umidade e temperatura.** 2019. Disponível em: < https://blog.eletrogate.com/sensores-dht11-dht22/#:~:text=O%20sensor%20DHT11%20%C3%A9%20um,umidade%20e%20temperatura%20do%20ar.&text=Dentro%20do%20sensor%20existe%20um,de%20um%20pino%20de%20sa%C3%ADda.>. Acesso em: 06 jun. 2021.

OLIVEIRA, Jailson. **ESP32 e suas versões.** 2019. Disponível em: <https://xprojetos.net/esp32-e-suas-versoes/#doit-esp32-devkit-v1>. Acesso em: 06 jun. 2021.

PINK, Farm. **Como Cultivamos**, 2020. Disponível em: <https://www.pinkfarms.com.br/sobre>. Acesso em: 06 jun. 2021.

QUINAGLIA, Augusto Carvajal. **MONITORAMENTO, COM ARMAZENAMENTO DE DADOS EM NUVEM, DAS VARIÁVEIS DE AMBIENTES DE ESTOQUE DE PEÇAS AERONÁUTICAS**. 2019.

SAVEH**. A disponibilidade de água no mundo e no Brasil,** 2016. Disponível em: <https://saveh.ambev.com.br/artigos/a-disponibilidade-de-agua-no-mundo-e-no-brasil#:~:text=Distribui%C3%A7%C3%A3o%20da%20%C3%A1gua%20no%20planeta%3A&text=Al%C3%A9m%20de%20ser%20um%20recurso,%2C%20%C3%8Dndia%2C%20Col%C3%B4mbia%20e%20Congo.>. Acesso em: 06 jun. 2021.

THOMSEN, Adilson. **Monitore sua planta.** 2016. Disponível em: <https://www.filipeflop.com/blog/monitore-sua-planta-usando-arduino/>. Acesso em: 06 jun. 2021.

TREENCE, Todd. **Adafruit IO basics feeds**, 2015. Disponível em: <https://learn.adafruit.com/adafruit-io-basics-feeds>. Acesso em: 06 jun. 2021.

TREENCE, Todd. **Adafruit IO basics dashboards**, 2015. Disponível em: <https://learn.adafruit.com/adafruit-io-basics-dashboards>. Acesso em: 06 jun. 2021.

VIDAL, Vitor. **Automação de sistema de irrigação: Sensor de umidade e válvula solenóide.** 2017. Disponível em: < http://blog.eletrogate.com/automacao-de-sistema-de-irrigacao-sensor-de-umidade-e-valvula-solenoide/>. Acesso em: 06 jun. 2021.

WRI. **Ranking the World's Most Water-Stressed Countries in 2040**, 2015. Disponível em: <https://www.wri.org/blog/2015/08/ranking-world-s-most-water-stressed-countries-2040>. Acesso em: 06 jun. 2021.

YASSEIN, Muneer Bani et al. Internet of Things: Survey and open issues of MQTT protocol. In: **2017 international conference on engineering & MIS (ICEMIS)**. IEEE, 2017. p. 1-6.

YUAN, Michael. **Conhecendo o MQTT.** 2017. Disponível em: <https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 06 jun. 2021

Anexo 1

```
/************************ Inclusão das Bibliotecas ********************************/
#include <DHT.h>
#include <WiFi.h>
#include <NTPClient.h> //https://github.com/arduino-libraries/NTPClient
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
 /************************ Conexão WiFi*********************************/
 const char* ssid = "------ocultado por motivos de segurança-----"; //Nome da rede
const char* password = "------ocultado por motivos de segurança-----"; //Senha
/********************** Credenciais Adafruit io ************************/
//Configurações como comunicação com Adafruit Nuvem
#define AIO_SERVER "io.adafruit.com"
#define AIO_SERVERPORT 1883 // use 8883 para SSL
#define AIO_USERNAME "------ocultado por motivos de segurança-----"
#define AIO_KEY "------ocultado por motivos de segurança-----"
/********************** Variaveis globais ******************************/
//-------- Configurações de relógio on-line----------
WiFiUDP udp;
NTPClient ntp(udp, "a.st1.ntp.br", -3 * 3600, 60000);//Cria um objeto "NTP" com as configurações.utilizada no Brasil
String hora;            // Variável que armazena
//Instancia a classeCliente
WiFiClient client;
//Configura a classe cliente MQTT informando os dados de Wifi, e do Servidor MQTT.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
//Ligação Física
#define DHTPIN1 27 //Dados do sensor 1
#define DHTPIN2 26 //Dados do sensor 2
#define DHTPIN3 25 //Dados do sensor 3
// Utilize a linha de acordo com o modelo do sensor
#define DHTTYPE DHT11 // Sensor DHT11
//#define DHTTYPE DHT22 // Sensor DHT 22 (AM2302)
//#define DHTTYPE DHT21 // Sensor DHT 21 (AM2301)
// Definicoes do sensor: pino, tipo
DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
DHT dht3(DHTPIN3, DHTTYPE);
#define SOLO1 32
#define SOLO2 34
#define SOLO3 35
#define AGUA 15
#define pino_ventilador 5
#define pinoRele 4
/*****************************Variageis globais usadas no processo*****************************/
long previousMillis = 0;
int i = 0;
float vazao; //Variável para armazenar o valor em L/min
float calibragem = 0.051546/5.5;
float media = 0;
int contaPulso = 0;; //Variável para a quantidade de pulsos
int s1 = 0;
int s2 = 0;
int s3 = 0;
float h1 = 0;
float h2 = 0;
float h3 = 0;
float h = 0;
float t1 = 0;
float t2 = 0;
float t3 = 0;
float t = 0;
```

```
/****************************** Declaração dos Topicos ************************************/
//Instancia-se classes para envio da informação no Feed do servidor broker Adafruit
  Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/temperatura", MQTT_QOS_1);
  Adafruit_MQTT_Publish umidade = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/umidade", MQTT_QOS_1);
  Adafruit_MQTT_Publish solo1 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/umidadesolo.vaso1", MQTT_QOS_1);
  Adafruit_MQTT_Publish solo2 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/umidadesolo.vaso2", MQTT_QOS_1);
  Adafruit_MQTT_Publish solo3 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/umidadesolo.vaso3", MQTT_QOS_1);
  //Adafruit_MQTT_Publish litros1 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/litros1", MQTT_QOS_1);
  //Adafruit_MQTT_Publish litros2 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/litros2", MQTT_QOS_1);
  //Adafruit_MQTT_Publish litros3 = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/litros3", MQTT_QOS_1);
  Adafruit_MQTT_Publish litros = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/litros", MQTT_QOS_1);
  Adafruit_MQTT_Publish status_ventilador = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/ventilador", MQTT_QOS_1);
  Adafruit_MQTT_Publish status_solenoide = Adafruit_MQTT_Publish(&mqtt,AIO_USERNAME"/feeds/solenoide", MQTT_QOS_1);
  Adafruit_MQTT_Subscribe rele_ventilador = Adafruit_MQTT_Subscribe(&mqtt,AIO_USERNAME"/feeds/ventilador", MQTT_QOS_1);
  Adafruit_MQTT_Subscribe rele_solenoide = Adafruit_MQTT_Subscribe(&mqtt,AIO_USERNAME"/feeds/solenoide", MQTT_QOS_1);
/***************************** Declaração dos Prototypes **********************************/
void initSerial();
void initPins();
void initWiFi();
void initRelogio();
void conectar_broker();
void atua(Adafruit_MQTT_Publish titulo);
void ventila(Adafruit_MQTT_Publish titulo);
void incpulso();
void escreveSerial(String titulo, float valor, String unidade);
void publica(Adafruit_MQTT_Publish titulo, float valor);
void publica_status(Adafruit_MQTT_Publish titulo, String valor);
void sensores();
void verifica();
void avalia_subscri();
/***************************** Sketch ************************************/
void setup() {
  initSerial();
  initPins();
  initWiFi();
  initRelogio();
  mqtt.subscribe(&rele_ventilador);
  mqtt.subscribe(&rele_solenoide);            }
void loop() {
  // ping adafruit io broker a few times to make sure we remain connected
  if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      conectar_broker();              }
  avalia_subscri();
  //Armazena na variável hora, o horário atual.
  hora = ntp.getFormattedTime();
  //Atua quando a umidade abaixa
  if (s1 <= 30 && s2 <= 30 && s3 <= 30 && hora >= "08:00:00" && hora <= "08:00:10"){
   atua(litros);
   delay(10000);       }
  //Atua quando a umidade abaixa
  /*if (s3 <= 30){
   atua(litros3);
   delay(10000);       }
  //Atua no horario 1
  if (hora >= "08:00:00" && hora <= "08:00:10" && s2 <= 30){
   atua(litros2);
   delay(10000);         }
  //Atua no horario 2
  if (hora >= "19:00:00" && hora <= "19:00:10"&& s1 <= 30){
   atua(litros1);
   delay(10000);         }*/
   // Função responsável por ler e enviar o valor do sensor a cada 30 segundos
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > 30000 && mqtt.connected()) {
```

```
    previousMillis = currentMillis;
    Serial.println(hora);     // Escreve a hora no monitor serial.
    sensores();
    verifica();
    publica(solo1, float(s1));
    publica(solo2, float(s2));
    publica(solo3, float(s3));
    publica(umidade, float(h));
    publica(temperatura, float(t));
    //Controle do ventilador
    if (t >= 24){
      digitalWrite(pino_ventilador, LOW);// Liga o ventilador;
      publica_status(status_ventilador, "ON");              }
    if (t <= 21){
      digitalWrite(pino_ventilador, HIGH);// Desliga o ventilador;
      publica_status(status_ventilador, "OFF");             } } }
/*************************** Implementação dos Prototypes ***********************************/
/* Conexao Serial */
void initSerial() {
  Serial.begin(115200);//BaudRate
  delay(10);             }
/* Configuração dos pinos */
void initPins() {
  // Inicializa os sensores de umidade do solo
  pinMode(SOLO1, INPUT);
  pinMode(SOLO2, INPUT);
  pinMode(SOLO3, INPUT);
  //DEFINE O PINO COMO SAÍDA
  pinMode(pino_ventilador, OUTPUT);
  // Ventilador INICIA DESLIGADO....(ligado no low)
  digitalWrite(pino_ventilador, HIGH);
  //DEFINE O PINO COMO SAÍDA
  pinMode(pinoRele, OUTPUT);
  //MÓDULO RELÉ INICIA DESLIGADO....(ligado no low)
  digitalWrite(pinoRele, HIGH );
  // Inicializa os sensores DHT
  dht1.begin();
  dht2.begin();
  dht3.begin();
  // Inicializa o sensor de vasao
  pinMode(AGUA, INPUT);     }
  attachInterrupt(digitalPinToInterrupt(AGUA), incpulso, FALLING); //Configura o pino AGUA(Interrupção 0) para trabalhar como interrupção
/* Configuração da conexão WiFi */
void initWiFi() {
   //Conecta o Wi-fi
  Serial.print("Conectando-se na rede ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  //Verifica se há internet conectada
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");                  }
  Serial.println();
  Serial.println("Conectado à rede com sucesso"); Serial.print("Endereço IP: "); Serial.println(WiFi.localIP()); }
 /* Inicia o Relogio */
void initRelogio() {
  ntp.begin();              // Inicia o protocolo
  ntp.forceUpdate();    // Atualização .        }
 /*************************** Implementação das Leituras e atuações***********************************/
/* Avalia os Subscritos */
void avalia_subscri(){
  Adafruit_MQTT_Subscribe *subscription;
  while (subscription = mqtt.readSubscription(5000)) {
    // Eventos do Ventilador
    if (subscription == &rele_ventilador) {
```

```
      // convert mqtt ascii payload to int
      char *value = (char *)rele_ventilador.lastread;
      Serial.print(F("Received: "));
      Serial.println(value);
      // Apply message to ventilador
      String message = String(value);
      message.trim();
      if (message == "ON") {digitalWrite(pino_ventilador, LOW);}
      if (message == "OFF") {digitalWrite(pino_ventilador, HIGH);}    }
    // Eventos da solenoide
    if (subscription == &rele_solenoide) {
      // convert mqtt ascii payload to int
      char *value = (char *)rele_solenoide.lastread;
      Serial.print(F("Received: "));
      Serial.println(value);
      // Apply message to soleoide
      String message = String(value);
      message.trim();
      if (message == "Open") {
        atua(litros);
        publica_status(status_solenoide, "Close");     }
      if (message == "Close") {digitalWrite(pinoRele, HIGH);}    }  }            }
/* atuação da irrigação via bomba ligada a Rele,
& leitura do fluxo de agua + chamada de publicação*/
void atua(Adafruit_MQTT_Publish titulo) {
  digitalWrite(pinoRele, LOW); // liga a agua
  publica_status(status_solenoide, "Open");
  //Faz a leitura 3 vezes
  i = 0; //Zera a variável i para uma nova contagem
  vazao = 0; //Zera a variável vazao para uma nova contagem
  media = 0; //Zera a variável media para uma nova contagem
  while (i < 2){
    contaPulso = 0;   //Zera a variável para contar os giros por segundos
    sei();      //Habilita interrupção
    delay (1000); //Aguarda 1 segundo
    cli();      //Desabilita interrupção
    i++;
    vazao = contaPulso*calibragem; //Converte para mL
    media=(media+vazao)/2; //Soma a vazão para o calculo da media
    escreveSerial("\nVazao injetada", vazao, "L");
    escreveSerial("", i, "s");  }
  digitalWrite(pinoRele, HIGH); //desliga a agua
  publica_status(status_solenoide, "Close");
   escreveSerial("\nMedia injetada = ", media, "L");
  publica(titulo, float(media));   }
//Incrementa a variável de contagem dos pulsos
void incpulso(){   contaPulso++; }
//Escreve no Monitor Serial
void escreveSerial(String titulo, float valor, String unidade){
  Serial.print(titulo); //Imprime a frase do titulo
  Serial.print(valor); //Imprime o valor
  Serial.println(unidade); //Imprime unidade
  Serial.println();}
//Publica os dados para o broker
void publica(Adafruit_MQTT_Publish titulo, float valor){
  if (! titulo.publish(float(valor))) {
      Serial.println(F("Failed"));    }
 else { Serial.println(F("OK!"));    } }
//Publica os dados para o broker
void publica_status(Adafruit_MQTT_Publish titulo, String valor){
  char Buf[10];
  valor.toCharArray(Buf, 50);
  if (! titulo.publish(Buf)) {
      Serial.println(F("Failed"));
    } else { Serial.println(F("OK!"));    }     }
```

```
//Faz a leitura dos sensores
void sensores()
{    // Leitura do solo1
   s1 = analogRead(32);
   // Leitura do solo 2
   s2 = analogRead(34);
   // Leitura do solo 3
   s3 = analogRead(35);


   // conversão da Leitura do solo1
   s1 = (-0.024420024*s1)+100;
   // conversão da Leitura do solo 2
   s2 = (-0.024420024*s2)+100;
   // conversão da Leitura do solo 3
   s3 = (-0.024420024*s3)+100;
   // Leitura da umidade1 (%)
   h1 = (float)dht1.readHumidity();
   // Leitura da umidade2 (%)
   h2 = (float)dht2.readHumidity();
   // Leitura da umidade3 (%)
   h3 = (float)dht3.readHumidity();
   h = (h1+h2+h3)/3;
     // Leitura da temperatura1 (Celsius)
   t1 = (float)dht1.readTemperature();
   // Leitura da temperatura2 (Celsius)
   t2 = (float)dht2.readTemperature();
   // Leitura da temperatura3 (Celsius)
   t3 = (float)dht3.readTemperature();
   t = (t1+t2+t3)/3;  }
//Verifica a leitura dos sensores e publica no Monitor Serial
void verifica()
{   // Verifica se o sensor1 SOLO está respondendo
 if (isnan(s1)){ //Caso o dado não seja válido
   Serial.println("Falha ao ler dados !!!");
   return;
 }else{ //Caso o dado seja válido mostra o dado no monitor
   escreveSerial("Umidade Solo 1: ", s1, " %");          }
 // Verifica se o sensor2 SOLO está respondendo
 if (isnan(s2)){ //Caso o dado não seja válido
   Serial.println("Falha ao ler dados !!!");
   return;
 }else{ //Caso o dado seja válido mostra o dado no monitor
   escreveSerial("Umidade Solo 2: ", s2, " %");          }
 // Verifica se o sensor3 SOLO está respondendo
 if (isnan(s3)){ //Caso o dado não seja válido
   Serial.println("Falha ao ler dados !!!");
   return;
 }else{ //Caso o dado seja válido mostra o dado no monitor
   escreveSerial("Umidade Solo 3: ", s3, " %");          }
 // Verifica se o sensor1 DHT11 está respondendo
 if (isnan(h1) || isnan(t1)){ //Caso o dado não seja válido
   Serial.println("Falha ao ler dados !!!");
   return;
 }else{ //Caso o dado seja válido mostra o dado no monitor
   escreveSerial("Temperatura1: ", t1, " *C ");
   escreveSerial("Umidade1 : ", h1, " %");  }
 // Verifica se o sensor2 DHT11 está respondendo
 if (isnan(h2) || isnan(t2)){ //Caso o dado não seja válido
   Serial.println("Falha ao ler dados !!!");
   return;
 }else{ //Caso o dado seja válido mostra o dado no monitor
   escreveSerial("Temperatura2: ", t2, " *C ");
   escreveSerial("Umidade2 : ", h2, " %");  }
 // Verifica se o sensor3 DHT11 está respondendo
```

```
  if (isnan(h3) || isnan(t3)){ //Caso o dado não seja válido
    Serial.println("Falha ao ler dados !!!");
    return;
  }else{ //Caso o dado seja válido mostra o dado no monitor
    escreveSerial("Temperatura3: ", t3, " *C ");
    escreveSerial("Umidade3 : ", h3, " %"); }
  // Verifica se a média é valida
  if (isnan(h)){ //Caso o dado não seja válido
    Serial.println("Falha ao relacionar dados !!!");
    return;
  }else{ //Caso o dado seja válido mostra o dado no monitor
    escreveSerial("Umidade Média: ", h, " %");              }
  // Verifica se a média é valida
  if (isnan(t)){ //Caso o dado não seja válido
    Serial.println("Falha ao relacionar dados !!!");
    return;
  }else{ //Caso o dado seja válido mostra o dado no monitor
    escreveSerial("Temperatura Média: ", t, " *C ");}    }
/************************** Demais implementações ***********************************/
 /* Conexão com o broker e também servirá para reestabelecer a conexão caso caia */
void conectar_broker() {
  Serial.print(F("Connecting to Adafruit broker mqtt... "));
  int8_t ret;
  while ((ret = mqtt.connect()) != 0) {
    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
      default: Serial.println(F("Connection failed")); break;
    }
    if(ret >= 0)
      mqtt.disconnect();
    Serial.println(F("Retrying connection..."));
    delay(5000);
  }
  Serial.println(F("Conectado ao broker com sucesso!"));
}
```