Universidade Federal de Ouro Preto
Escola de Minas
CECAU - Colegiado do Curso de
Engenharia de Controle e Automação

Davi Pereira Hilal

Desenvolvimento de um Gateway IoT para um Sistema de Monitoramento e Controle de Irrigação

Monografia de Graduação

Ouro Preto, 2025

Davi Pereira Hilal

Desenvolvimento de um Gateway IoT para um Sistema de Monitoramento e Controle de Irrigação

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro(a) de Controle e Automação.

Universidade Federal de Ouro Preto

Orientador: Prof. Alan Kardek Rêgo Segundo, D.Sc.

Ouro Preto

2025

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

H641d Hilal, Davi Pereira.
Desenvolvimento de um Gateway IoT para um sistema de monitoramento e controle de irrigação. [manuscrito] / Davi Pereira Hilal. - 2025.
50 f.: il.: color., gráf..

Orientador: Prof. Dr. Alan Kardek Rêgo Segundo.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. CGI (Protocolo de rede de computador). 2. Internet das coisas. 3. Microcontroladores - ESP8266. 4. Message Queuing Telemetry Transport (MQTT). 5. Sistemas operacionais (Computadores) - Over-The-Air (OTA). 6. Agronomia - Automação. 7. Inovações agrícolas. I. Rêgo Segundo, Alan Kardek. II. Universidade Federal de Ouro Preto. III. Título.

CDU 681.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Davi Pereira Hilal

Desenvolvimento de um Gateway IoT para um Sistema de Monitoramento e Controle de Irrigação

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 11 de fevereiro de 2025

Membros da banca

Dr. Alan Kardek Rêgo Segundo - Orientador (Universidade Federal de Ouro Preto)
Dr. Agnaldo José da Rocha Reis - Convidado (Universidade Federal de Ouro Preto)
Dr. Paulo Marcos de Barros Monteiro - Convidado (Universidade Federal de Ouro Preto)

Alan Kardek Rêgo Segundo, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 13/02/2025

Documento assinado eletronicamente por **Alan Kardek Rego Segundo**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 13/02/2025, às 10:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0856312** e o código CRC **9959766D**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Resumo

A modernização da agricultura e a busca por maior eficiência no uso da água têm impulsionado o desenvolvimento de novas tecnologias para automação de processos, como a irrigação. Neste contexto, este trabalho apresenta o desenvolvimento e a implementação de um dispositivo de comunicação, conhecido como *gateway* IoT, capaz de integrar um sistema de irrigação automatizado à internet. O dispositivo foi construído utilizando o ESP8266, um SoC (System on a Chip) compacto com conectividade Wi-Fi, e o protocolo MQTT (Message Queuing Telemetry Transport), um padrão leve e eficiente para a troca de mensagens entre dispositivos conectados. Além de coletar e transmitir dados das estações de medição de umidade do solo e do controlador de irrigação, o *gateway* também possibilita atualizações remotas de *firmware* via OTA (Over-the-Air), um processo que permite instalar novas versões do software sem necessidade de acesso físico ao dispositivo. Para validar sua funcionalidade, o sistema foi testado em um ambiente real, onde demonstrou eficiência na recepção e envio de dados, além de permitir a manutenção remota. Durante os testes, a taxa média de perda de pacotes foi de 7,195%, o que não comprometeu a confiabilidade das medições diárias, uma vez que a maior parte perdas não ocorreu de forma consecutiva. Isso garante a confiabilidade dos dados e a eficiência do sistema de monitoramento.

Palavras-chaves: *Gateway*, IoT, ESP8266, MQTT, Atualização OTA, Automação agrícola.

Abstract

The modernization of agriculture and the pursuit of greater efficiency in water use have driven the development of new technologies for process automation, such as irrigation. In this context, this work presents the development and implementation of a communication device, known as an IoT gateway, capable of integrating an automated irrigation system with the internet. The device was built using the ESP8266, a compact SoC (System on a Chip) with Wi-Fi connectivity, and the MQTT (Message Queuing Telemetry Transport) protocol, a lightweight and efficient standard for exchanging messages between connected devices. In addition to collecting and transmitting data from soil moisture monitoring stations and the irrigation controller, the gateway also enables remote firmware updates via OTA (Over-the-Air), a process that allows installing new software versions without the need for physical access to the device. To validate its functionality, the system was tested in a real environment, where it demonstrated efficiency in data reception and transmission, as well as allowing remote maintenance. During the tests, the average packet loss rate was 7.195%, which did not compromise the reliability of daily measurements, as most parts of the losses did not occur consecutively. This ensures the reliability of the data and the efficiency of the monitoring system.

Key-words: *Gateway*, IoT, ESP8266, MQTT, OTA Update, Agricultural Automation.

Lista de ilustrações

Lista de abreviaturas e siglas

ADC	Analog to Digital Conversion
AP	Access Point
AWS	Amazon Web Services
ESP	Espressif Systems (ESP8266, ESP32)
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
JSON	JavaScript Object Notation
LPWAN	Low Power Wide Area Network
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
OTA	Over-the-Air
QoS	Quality of Service
RF	Radio Frequency
RFID	Radio Frequency Identification
SoC	System on a Chip
SPI	Serial Peripheral Interface
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WDT	Watchdog Timer
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks
XCTU	XBee Configuration and Test Utility

Sumário

1 Introdução

Com o desenvolvimento de sensores, microchips e capacidades de análise de dados cada vez mais avançados, a capacidade de observar ambientes e entender estes tem se expandindo rapidamente. Esses dispositivos, que variam desde sistemas simples de monitoramento e fluxos de dados, até sofisticados sensores biosensoriais, estão mudando a forma como as máquinas interagem com o ambiente e como os seres humanos se conectam uns com os outros (GREENGARD, 2015).

A Internet das Coisas (IoT) é descrita como uma rede de dispositivos computacionais, máquinas digitais e mecânicas, objetos, seres vivos, ou até mesmo pessoas, todos interconectados e identificados de forma única. Esses componentes são capazes de comunicar e compartilhar dados através de uma rede sem a necessidade de intervenção humana, seja na interação direta com outros humanos ou com sistemas computacionais (ELIJAH et al., 2018). A Internet das Coisas surge como uma integração de várias tecnologias existentes, como redes de sensores sem fio (WSN), identificação por radiofrequência (RFID), computação em nuvem e sistemas de *middleware*, proporcionando uma comunicação eficiente entre o mundo físico e o digital (QUY et al., 2022). Esses dispositivos capturam dados do ambiente, que são processados e disponibilizados em interfaces de fácil acesso, otimizando a eficiência operacional, como por exemplo, da produtividade agrícola (ELIJAH et al., 2018).

O uso de sistemas inteligentes de irrigação, baseados em IoT, tem mostrado eficácia em reduzir o desperdício de água e aprimorar a eficiência na gestão hídrica, ajudando a evitar perdas e garantindo que as plantas recebam a quantidade exata de água necessária para seu crescimento, o que contribui para uma agricultura mais sustentável (XU; GU; TIAN, 2022).

Um *gateway* de IoT é um dispositivo de hardware específico que atua como intermediário entre os dispositivos e a rede, traduzindo dados de protocolos de comunicação de curta distância para redes convencionais. A principal dificuldade no desenvolvimento de um *gateway* IoT reside na falta de padronização, uma vez que cada dispositivo pode operar com um protocolo distinto, frequentemente incompatível com os outros, o que torna desafiadora a criação de um *gateway* universal. Apesar disso, todos os *gateways* devem atender a requisitos fundamentais, como baixo custo de hardware, facilidade de implementação, extensibilidade e suporte à camada de aplicação (GLÓRIA; CERCAS; SOUTO, 2017).

Neste trabalho, é apresentado o desenvolvimento de um *gateway* IoT baseado no SoC (System on a Chip) ESP8266, destinado ao monitoramento e controle remoto de sistemas

de irrigação automatizados. O *gateway* foi projetado para integrar dados de estações de monitoramento de umidade do solo e controladores de irrigação, permitindo a transmissão eficiente dessas informações e a atualização remota do *firmware* dos dispositivos. É detalhado a arquitetura do sistema, os protocolos de comunicação utilizados e as estratégias implementadas para permitir escalabilidade e a eficiência da dispositivo proposto.

1.1 Justificativas e Relevância

O desenvolvimento de sistemas automatizados e monitorados à distância é uma tendência crescente em diversas áreas, e o setor agrícola, especialmente no que diz respeito à irrigação inteligente, tem se beneficiado dessas inovações. A crescente demanda por eficiência no uso dos recursos hídricos e a necessidade de reduzir custos operacionais impulsionam a adoção de tecnologias como a Internet das Coisas para o controle e monitoramento remoto de sistemas de irrigação. Nesse contexto, o presente trabalho visa a criação de um *gateway* IoT, que se mostra necessário para integrar diferentes dispositivos de monitoramento e controle de uma maneira eficiente e de fácil manejo.

A relevância deste projeto reside na capacidade de o *gateway* proporcionar monitoramento em tempo real, permitindo aos usuarios acompanharem o funcionamento do sistema de irrigação. Além disso, um sistema que permite atualizações OTA (Over-the-Air), elimina a necessidade de intervenções físicas para manutenção ou melhorias no *firmware* dos dispositivos.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo principal desenvolver e implementar um *gateway* IoT para o monitoramento e controle remoto de um sistema de irrigação automatizado.

1.2.2 Objetivos Específicos

1. Avaliar a eficiência do *gateway* na recepção e retransmissão de mensagens das estações de monitoramento e do controlador de irrigação para um servidor via protocolo MQTT.

2. Analisar a confiabilidade do sistema de atualização remota do *firmware* do *gateway*, verificando sua robustez e possíveis falhas durante o processo.

3. Avaliar o funcionamento do sistema em um ambiente real de irrigação agrícola, testando a comunicação e a eficiência das atualizações remotas.

1.3 Estrutura do Trabalho

Este trabalho está dividido em cinco capítulos.

No **Capítulo 1** - Introdução, são apresentados o contexto, a motivação e os objetivos do estudo, destacando a importância da automação na agricultura e o desenvolvimento do gateway IoT.

O **Capítulo 2** - Revisão de Literatura aborda os conceitos e tecnologias essenciais para o projeto, como os protocolos de comunicação MQTT e HTTP/HTTPS, o funcionamento de microcontroladores e a integração de dispositivos IoT com redes sem fio.

No **Capítulo 3** - Desenvolvimento, são detalhados o funcionamento do sistema, a configuração do hardware e software, o fluxo de comunicação e a implementação da atualização OTA.

O **Capítulo 4** - Resultados apresenta os testes realizados, avaliando o desempenho do gateway, a confiabilidade do monitoramento remoto e o impacto da perda de pacotes em um ambiente real.

Por fim, o **Capítulo 5** - Conclusão resume os principais achados do estudo, destacando suas contribuições e possíveis melhorias futuras.

2 Revisão de literatura

O referencial teórico explora as principais tecnologias e conceitos que fundamentam o desenvolvimento de um gateway IoT para monitoramento e controle de sistemas de irrigação. São abordados os protocolos de comunicação, como MQTT e HTTP/HTTPS. Além disso, o uso de dispositivos, como o ESP8266, e módulos de comunicação, como o XBee. Também são apresentados conceitos de segurança e atualização remota de *firmware* via OTA.

2.1 HTTP e HTTPS

O HTTP (Hypertext Transfer Protocol) é o protocolo fundamental utilizado para comunicação na web, responsável pela troca de informações entre clientes, como navegadores, e servidores. Ele opera em um modelo de requisição-resposta, onde o cliente faz uma solicitação e o servidor retorna os dados requisitados. Contudo, o HTTP é um protocolo sem segurança nativa, o que significa que todas as informações transmitidas são enviadas em texto simples, tornando-as vulneráveis a interceptações. Isso é uma grande preocupação em ambientes onde dados sensíveis, como senhas ou números de cartões de crédito, são transferidos (GOURLEY; TOTTY, 2002).

Para solucionar essa falha de segurança, foi desenvolvido o HTTPS (Hypertext Transfer Protocol Secure). O HTTPS combina o protocolo HTTP com uma camada adicional de segurança, que é implementada por meio do SSL (Secure Sockets Layer) ou do TLS (Transport Layer Security). Esses protocolos de segurança criptografam as informações transmitidas entre o cliente e o servidor, garantindo que os dados não possam ser acessados por terceiros durante o trânsito na rede (SHKLAR; ROSEN, 2003).

O processo de criptografia é iniciado através de um “handshake”, no qual o cliente e o servidor trocam informações criptográficas para estabelecer uma conexão segura. Durante essa etapa, o servidor envia um certificado digital, emitido por uma autoridade certificadora (CA), que verifica sua identidade. Esse certificado assegura ao cliente que está se conectando ao servidor correto, protegendo contra ataques de intermediários, como o ”man-in-the-middle”, onde um invasor poderia interceptar a comunicação sem que o usuário ou o servidor percebam (GOURLEY; TOTTY, 2002).

O uso do HTTPS é particularmente importante em situações que envolvem a transmissão de informações sensíveis, como em transações bancárias, compras online e sistemas de login. Além de proteger a confidencialidade das informações transmitidas, o HTTPS garante a integridade dos dados, evitando que sejam modificados durante a

transmissão (SHKLAR; ROSEN, 2003).

2.2 MQTT (Message Queuing Telemetry Transport)

O MQTT (Message Queuing Telemetry Transport) é um protocolo leve de comunicação, amplamente utilizado em aplicações de Internet das Coisas e sistemas *Machine-to-Machine* (M2M). Ele é baseado no modelo de comunicação *publish/subscribe* e foi projetado para operar em redes com largura de banda limitada e conexões intermitentes, características comuns em sistemas IoT (MISHRA; KERTESZ, 2020). Este protocolo permite que dispositivos de baixo poder computacional e baixa energia se comuniquem de forma eficiente, transmitindo mensagens entre sensores e atuadores em uma rede.

O modelo publish/subscribe que o MQTT utiliza permite a comunicação de dispositivos de forma escalável. Neste modelo, um dispositivo pode publicar dados (publisher) em um tópico específico no servidor *broker*, enquanto outros dispositivos podem se inscrever (subscribe) nesse tópico para receber as atualizações, como representado na Figura 1. O *broker* MQTT é responsável por gerenciar essa comunicação, garantindo que as mensagens publicadas sejam distribuídas aos assinantes de acordo com suas devidas inscrições.

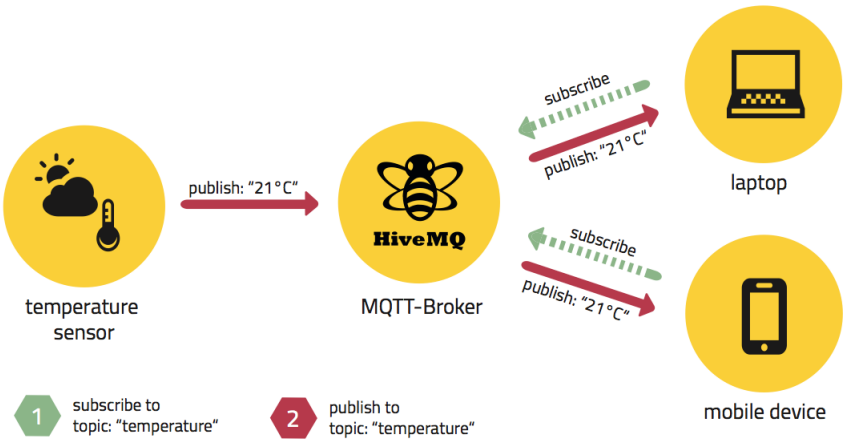

Figura 1 – Representação MQTT. Fonte: https://www.hivemq.com.

Uma das vantagens do MQTT é sua capacidade de operar em redes com baixa confiabilidade e alta latência. Para garantir a entrega das mensagens, o protocolo oferece três níveis de Qualidade de Serviço (QoS), que variam de acordo com a criticidade dos dados transmitidos. O QoS 0, chamado de “No máximo uma vez”, garante a entrega da mensagem uma vez sem confirmação de recebimento. Já o QoS 1, ”pelo menos uma vez”, garante que a mensagem será entregue pelo menos uma vez, podendo ser duplicada. O QoS 2, ”exatamente uma vez”, assegura que a mensagem será entregue exatamente uma

vez, sendo o nível mais confiável, mas com maior sobrecarga no processo de transmissão (MISHRA; KERTESZ, 2020).

O MQTT é ideal para aplicações que exigem baixo consumo de energia e comunicação assíncrona, como monitoramento remoto de sensores em áreas rurais ou aplicações industriais onde a confiabilidade e a simplicidade são essenciais. Por exemplo, em uma aplicação típica de monitoramento de temperatura, um sensor de temperatura pode publicar a leitura de 21°C para um tópico "temperature" no *broker* MQTT. Dispositivos como laptops ou smartphones que estão inscritos neste tópico recebem essa informação e podem tomar decisões ou acionar outros dispositivos com base nesses dados.

A flexibilidade do MQTT se deve à sua compatibilidade com uma variedade de dispositivos. Ele também suporta criptografia SSL/TLS para garantir a segurança dos dados transmitidos, tornando-o apropriado para uma ampla gama de aplicações sensíveis à segurança, como sistemas de saúde e automação residencial (HILLAR, 2017).

2.3 Microcontroladores

Os microcontroladores surgiram como uma evolução dos microprocessadores, especialmente desenhados para realizar tarefas de controle em sistemas embarcados. Ao contrário dos microprocessadores, que são projetados para realizar grandes cálculos, os microcontroladores têm a função de executar ações específicas em dispositivos eletrônicos. Com a integração de CPU, memória e periféricos em um único chip, eles são ideais para sistemas que requerem controle preciso, como automação e dispositivos eletrônicos do dia a dia (WILMSHURST, 2009).

Os microcontroladores incluem uma variedade de periféricos, como portas de entrada e saída (GPIO), temporizadores, conversores analógico-digital (ADC) e módulos de comunicação, como SPI e I^2C. Esses periféricos são essenciais para conectar o microcontrolador a sensores, atuadores e outros dispositivos. A correta utilização dos periféricos permite a criação de sistemas que monitoram e controlam variáveis físicas (VALDES-PEREZ; HERNANDEZ-AVELAR, 2009).

Além de periféricos como ADCs e temporizadores, os microcontroladores modernos incluem recursos que permitem o controle de sistemas em tempo real. Por exemplo, temporizadores são usados para controlar a duração de eventos e interrupções são gerenciadas para responder rapidamente a estímulos externos. Em sistemas onde a latência e o tempo de resposta são críticos, como em aplicações industriais ou médicas, esses recursos são necessários. A flexibilidade de configurar e utilizar esses periféricos permite que os microcontroladores sejam amplamente adaptados para diferentes tipos de sistemas embarcados (VALVANO, 2012).

Em muitos projetos práticos, como os descritos por (SCHWARTZ, 2016), o SoC acessível ESP8266 tem sido amplamente utilizado devido à sua capacidade de integrar Wi-Fi e controle de dispositivos em um único chip. Exemplos de uso incluem sistemas de automação residencial, controle de lâmpadas e sensores, além de aplicações mais complexas como robôs controlados via internet e plataformas de monitoramento de condições ambientais. Esses exemplos demonstram a flexibilidade dos SoCs atuais, que combinam microcontrolador e outros componentes, permitindo atender a vários tipos de situações.

2.4 Wi-Fi e Access Point

O Wi-Fi é uma tecnologia de comunicação sem fio baseada no padrão IEEE 802.11, que permite a troca de dados entre dispositivos. Essa tecnologia opera em bandas de frequência de 2.4 GHz ou 5 GHz, proporcionando flexibilidade e alcance adequados para diversos tipos de dispositivos, desde laptops até sistemas embarcados (GAST, 2005). No ambiente de redes locais, os dispositivos Wi-Fi se conectam a um Access Point, que funciona como uma ponte entre os dispositivos sem fio e a rede com fio, facilitando a comunicação e gerenciamento do tráfego de dados (TANENBAUM, 2011).

No contexto dos SoCs, como o ESP8266, o Access Point é uma funcionalidade importante. Ao operar como um ponto de acesso, o microcontrolador pode criar sua própria rede Wi-Fi, permitindo que dispositivos como smartphones e laptops se conectem diretamente a ele. Isso se torna útil durante o processo de configuração inicial do dispositivo, onde por exemplo, o usuário precisa inserir informações, como o nome da rede e a senha, que pode ser feito diretamente através de uma interface web local hospedada na mesma rede. Essa funcionalidade elimina a necessidade de um roteador externo e oferece uma solução eficiente para sistemas de Internet das Coisas que precisam ser configurados (KOLBAN, 2017).

2.5 Webserver

Um Webserver é um software que responde a requisições HTTP enviadas por clientes, como navegadores web. Ele é responsável por processar as requisições e enviar de volta respostas, que podem incluir páginas HTML ou arquivos específicos. O webserver atua como intermediário entre o cliente e o servidor, facilitando a transferência de informações e a visualização de conteúdos pela web (TANENBAUM, 2011). Quando o cliente faz uma solicitação, o servidor processa a requisição e retorna a resposta apropriada.

No SoC ESP8266, a implementação de um webserver local permite que o dispositivo ofereça uma interface gráfica para os usuários. Através de um navegador, é possível acessar o webserver para configurar o dispositivo, monitorar seus dados ou controlar suas funções

remotamente. A habilidade de um dispositivo embarcado rodar um webserver e processar requisições HTTP diretamente aumenta significativamente sua versatilidade e integração em redes locais ou sistemas distribuídos (KOLBAN, 2017).

2.6 WeMos D1 Mini (ESP8266)

O ESP8266 é um SoC popular em projetos de Internet das Coisas devido ao seu baixo custo e alta capacidade de conectividade sem fio. Fabricado pela Espressif Systems, opera na faixa de 2,4 GHz e segue os padrões IEEE 802.11 b/g/n, o que permite que dispositivos conectados a ele façam parte de uma rede Wi-Fi (ESPRESSIF SYSTEMS, 2020). O ESP8266 possui um conjunto robusto de interfaces, incluindo GPIOs, SPI, I2C, PWM e UART, possibilitando a integração com uma vasta gama de sensores e atuadores (ESPRESSIF SYSTEMS, 2020). Um ponto notável desse dispositivo é a sua capacidade de entrar em modos de economia de energia, como o modo deep sleep, onde o consumo de energia pode chegar de 8 à 20µA, sendo extremamente baixo, o que o torna ideal para aplicações alimentadas por baterias (ESPRESSIF SYSTEMS, 2020).

Além disso, o ESP8266 é utilizado em diversas plataformas de prototipagem, sendo uma das mais conhecidas a WeMos D1 Mini, que integra o ESP8266 em um formato mais acessível para desenvolvedores. O WeMos, além de facilitar o desenvolvimento com ESP8266, oferece uma interface USB para programação e alimentação, e disponibiliza diversos pinos GPIO adicionais. Na prática, o uso do WeMos oferece todas as funcionalidades do ESP8266 com uma camada extra de conveniência para quem desenvolve projetos IoT, permitindo rápida prototipagem e fácil acesso a todas as suas funcionalidades.

Os módulos ESP8266 suportam múltiplos modos de comunicação, incluindo o modo estação (station) e o modo ponto de acesso (access point), podendo inclusive operar em modo dual, o que permite ao dispositivo atuar como cliente e servidor em uma rede Wi-Fi simultaneamente. Esse recurso é amplamente explorado em soluções IoT que exigem comunicação direta entre dispositivos e uma interface de rede centralizada, como gateways e hubs (ESPRESSIF SYSTEMS, 2020).

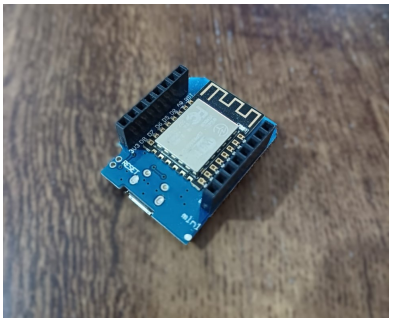

Figura 2 – WeMos D1 Mini. Fonte: De autoria própria

2.6.1 OTA para o ESP8266

A atualização OTA para o ESP8266 permite a reprogramação remota do *firmware* enquanto o dispositivo continua operando normalmente. O sistema pode ser atualizado utilizando a conectividade do dispositivo, como o *WiFi*, sem interromper o funcionamento do *firmware* em execução (ESPRESSIF SYSTEMS, 2024). Isso é especialmente útil em dispositivos IoT localizados em áreas de difícil acesso, facilitando a manutenção e permitindo melhorias contínuas no software.

No processo de OTA, o ESP8266 utiliza uma técnica de gerenciamento de memória flash que permite armazenar duas imagens de *firmware*: a imagem atual em execução e uma nova imagem que será carregada durante a atualização. Essa abordagem de particionamento de memória, conhecida como dual-partition OTA, permite que o ESP8266 valide a nova imagem antes de substituí-la, garantindo que o dispositivo possa reverter para a versão anterior em caso de falha durante a atualização (ESPRESSIF SYSTEMS, 2024).

A OTA via HTTP é o método mais simples, onde o ESP8266 baixa o novo *firmware* de um servidor HTTP e o armazena na memória flash (COMMUNITY, 2024). O processo pode ser iniciado programaticamente pelo dispositivo ou em resposta a um comando externo. No entanto, esse método pode ser vulnerável, já que os dados são transmitidos sem criptografia, expondo-os a ataques de interceptação.

Para aplicações que exigem maior segurança, o HTTPS é a solução ideal. Utilizando criptografia, o HTTPS protege os dados transmitidos e garante que o *firmware* não seja interceptado ou adulterado. O uso de certificados digitais permite ao ESP8266 validar a autenticidade do servidor antes de baixar o novo *firmware*. Para isso, o dispositivo precisa de um certificado raiz embutido, que valida a cadeia de confiança do certificado do servidor. Embora o HTTPS consuma mais recursos e seja mais complexo de configurar, ele é essencial para garantir a integridade e autenticidade do *firmware* em aplicações críticas.

2.7 Módulo XBee

O XBee é um módulo de comunicação sem fio amplamente utilizado em projetos de Internet das Coisas devido à sua confiabilidade e eficiência. Como mostrado na Figura 3, o módulo XBee Pro, fabricado pela Digi International, é um exemplo de hardware robusto que suporta a comunicação baseada no protocolo Zigbee, permitindo a criação de redes de malha. Este módulo é compatível com uma ampla gama de dispositivos, o que facilita sua integração em diferentes projetos, desde aplicações simples até sistemas mais complexos.

O XBee oferece dois modos de operação distintos, cada um atendendo a diferentes necessidades de projeto. No modo transparente, como ilustrado na Figura 4, o XBee funciona como uma substituição direta do cabo serial, transmitindo os dados recebidos

Figura 3 – Módulo XBee. Fonte: De autoria própria

sem qualquer modificação ou formatação. Este modo é ideal para aplicações que exigem uma implementação rápida e sem complexidades adicionais, tornando a comunicação entre dispositivos simples e direta.

Figura 4 – Representação modo Transparente. Fonte: (DIGI INTERNATIONAL, 2024).

Para projetos que requerem maior controle e flexibilidade, o XBee pode ser configurado para operar no modo API, conforme mostrado na Figura 5. Neste modo, os dados são encapsulados em pacotes, permitindo o gerenciamento de múltiplos dispositivos, o endereçamento dinâmico, e a transmissão de dados com maior segurança e confiabilidade. Essa funcionalidade é crucial para aplicações que exigem um controle mais preciso sobre a comunicação e a estrutura da rede.

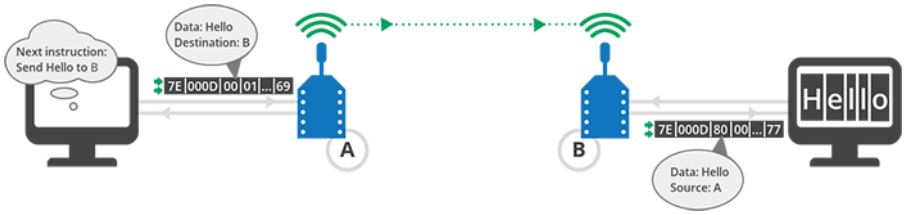

Figura 5 – Representação modo API. Fonte: (DIGI INTERNATIONAL, 2024).

O módulo XBee opera na faixa de frequência ISM de 2,4 GHz e é eficiente para redes de sensores de baixo custo e baixa potência. Ele possui variantes como XBee Série 1, XBee-PRO Série 1, XBee Série 2 e XBee-PRO Série 2, todas compatíveis entre si e adequadas para diferentes ambientes.O XBee Série 1 tem alcance de até 30 metros em ambientes internos e até 90 metros em ambientes externos. O XBee-PRO Série 1 alcança até

90 metros em ambientes internos e até 1,6 km em ambientes externos (XBEE/XBEE-PRO..., 2009). Já o XBee Série 2 alcança até 40 metros em ambientes internos e 120 metros em ambientes externos. O XBee-PRO Série 2 alcança até 90 metros em ambientes internos e até 3,2 km em ambientes externos (DIGI INTERNATIONAL, 2022).

Ambos os módulos possuem uma taxa de dados RF de 250 kbps e excelente sensibilidade de recepção, com -92 dBm para o XBee e -100 dBm para o XBee-PRO, o que garante a integridade dos dados transmitidos. Em termos de consumo de energia, o XBee consome 45 mA em modo de transmissão e 50 mA em modo de recepção, enquanto o XBee-PRO consome 250 mA e 55 mA, respectivamente. Ambos os módulos podem entrar em modo de hibernação com consumo inferior a 10 µA, o que os torna ideais para aplicações que exigem baixo consumo de energia (DIGI INTERNATIONAL, 2018).

Além disso, esses módulos suportam várias topologias de rede, como ponto a ponto, ponto a multiponto e peer-to-peer. A segurança e confiabilidade da comunicação são asseguradas pela retransmissão e confirmação de pacotes, bem como pela tecnologia DSSS (Direct Sequence Spread Spectrum), que reduz a interferência e melhora a integridade dos dados transmitidos (DIGI INTERNATIONAL, 2018).

2.8 UART (Universal Asynchronous Receiver/Transmitter)

A UART (Universal Asynchronous Receiver/Transmitter) é um componente fundamental em sistemas embarcados e de comunicação serial assíncrona. Sua principal função é converter dados paralelos em série e vice-versa, possibilitando a troca de informações entre dispositivos como microcontroladores, PCs e periféricos. O UART é amplamente utilizado devido à simplicidade e à economia de recursos em comparação com outros métodos de comunicação, como interfaces síncronas (HOROWITZ; HILL, 2015).

A comunicação UART segue um protocolo assíncrono, no qual não há linha de clock compartilhada entre os dispositivos comunicantes. Em vez disso, cada dispositivo utiliza sua própria referência de tempo, e o sincronismo é obtido através de bits de início e parada, além de uma taxa de transmissão (bit rate) comum entre os dispositivos (AXELSON, 2019). O UART envia dados em pacotes que incluem um bit de início, 5 a 8 bits de dados, um bit opcional de paridade e um ou dois bits de parada. Essa flexibilidade permite que o UART seja configurado para diferentes tipos de comunicação, dependendo da aplicação (AXELSON, 2019).

Uma das grandes vantagens do UART é a sua compatibilidade com várias arquiteturas de microcontroladores. Devido à sua simplicidade, ele é comumente encontrado em dispositivos de 8, 16 e 32 bits. A UART é ideal para aplicações de baixa velocidade e longas distâncias, como em redes RS-232 e RS-485, que suportam transmissões de até 4000 pés em alguns casos, tornando-o uma escolha viável em ambientes industriais e de

monitoramento (AXELSON, 2019).

Embora muitos sistemas modernos utilizem USB ou Ethernet para a comunicação de alta velocidade, a UART continua sendo amplamente adotada, especialmente em sistemas embarcados, onde o baixo custo, a baixa complexidade e o uso de cabos longos são fatores essenciais. Além disso, conversores USB para serial permitem que PCs modernos, que geralmente não possuem portas seriais nativas, se comuniquem com dispositivos UART, mantendo a relevância dessa tecnologia em uma ampla gama de aplicações (AXELSON, 2019).

Para sistemas embarcados que exigem mais flexibilidade, alguns microcontroladores oferecem USART (Universal Synchronous/Asynchronous Receiver/Transmitter), que adiciona a capacidade de comunicação síncrona, permitindo maior controle sobre a sincronização dos dados transmitidos (HOROWITZ; HILL, 2015).

2.9 Trabalhos relacionados

Para identificar estudos correlatos com este projeto, foram utilizadas as seguintes regras de pesquisa na plataforma CAPES: palavras-chave como “IoT, gateway, irrigation, automation”; acesso aberto; artigos publicados entre 2020 e 2025; na área de Engenharias.

Froiz-Míguez et al. (FROIZ-MÍGUEZ et al., 2020) apresentaram um sistema de irrigação inteligente baseado em LoRa e LoRaWAN, que visa cobrir grandes áreas urbanas e suburbanas utilizando comunicações LPWAN (Low Power Wide Area Network). O sistema faz uso de computação em névoa (fog computing) para processar dados localmente e otimizar o consumo de energia e a cobertura da rede de sensores. O trabalho destaca a importância da seleção adequada das localizações dos gateways para reduzir o consumo de energia e melhorar a conectividade.

Kilaru et al. (KILARU et al., 2022) propuseram um sistema automatizado de irrigação utilizando Redes de Sensores Sem Fio (WSN) e previsão meteorológica para otimizar o uso de água. A topologia utilizada é em forma de árvore, o que permite aumentar o alcance e facilitar a integração de novos nós à rede.

Lou et al. (LOU et al., 2020) desenvolveram um sistema de monitoramento do ambiente agrícola utilizando NB-IoT, voltado para a redução do desperdício de água e a simplificação da infraestrutura de comunicação. O sistema proposto utiliza sensores para monitorar a umidade do solo, nutrientes e parâmetros ambientais, transmitindo os dados em tempo real através do NB-IoT. Essa abordagem pode ser uma alternativa interessante para ambientes que necessitem de longa distância de comunicação.

Murali e Sridhar (MURALI; SRIDHAR, 2021) propuseram um sistema de irrigação inteligente que utiliza arrays de sensores e comunicação via ESP8266. A proposta traz ideias

úteis para a implementação de soluções baseadas em sensores distribuídos e comunicação Wi-Fi. Além disso, o uso de ESP8266 mostrou-se uma alternativa de baixo custo e que possibilita integração com sistemas de controle e monitoramento de forma simples e barata.

Habib et al. (HABIB et al., 2023) desenvolveram um sistema de irrigação automatizado baseado em IoT, utilizando água residual como fonte para a irrigação. O sistema permite o controle remoto das bombas de água e o monitoramento de variáveis como umidade e vazão, utilizando um dashboard HTTP para visualização dos dados.

Wardana et al. (WARDANA et al., 2018/12) apresentaram um sistema de monitoramento e controle para irrigação por gotejamento utilizando NodeMCU e Raspberry Pi. O sistema faz uso do protocolo MQTT para a comunicação entre sensores e atuadores, permitindo uma transmissão de dados eficiente e em tempo real. A utilização do MQTT Broker instalado no Raspberry Pi com a plataforma Mosquitto fornece um ambiente leve e adequado para aplicações IoT.

Após a revisão da literatura em busca de soluções para sistemas de irrigação automatizada baseados em IoT, verificou-se que, na maioria dos trabalhos, o foco principal está na integração de sensores e atuadores para o monitoramento de variáveis ambientais e controle das operações de irrigação. No entanto, poucos estudos tratam da manutenção dos dispositivos IoT em campo.

3 Desenvolvimento

Neste capítulo, é apresentado o sistema de monitoramento de umidade de solo e controle de irrigação oferecido pela Epsilon Automação[1], como é feita a troca de mensagens e qual topologia de rede é utilizada para isso. O desenvolvimento do *gateway*, que integra o sistema e permite a conexão com a internet, também é discutido, incluindo detalhes sobre seu hardware e software. O capítulo aborda como o *gateway* gerencia a comunicação entre os dispositivos em campo e realiza o envio de dados para um servidor remoto, além de integrar a funcionalidade de atualização remota de *firmware* (OTA).

3.1 Sistema de Aquisição e Controle de Irrigação

3.1.1 Estação de Monitoramento de Solo

Inicialmente, para se ter uma ideia de como cada parte do sistema funciona, é importante entender o papel da estação de monitoramento de solo. Ela é responsável por coletar dados do solo para a automação da irrigação, garantindo que as decisões de irrigar sejam baseadas em medições atuais. O sistema conta com 4 sensores capacitivos que medem a umidade, a salinidade e a temperatura do solo, proporcionando que seja feita uma análise das condições em campo.

A estação é completamente autônoma, alimentada por uma bateria de 12 V recarregada por um painel solar, o que elimina a necessidade de manutenções frequentes ou fontes externas de energia. A transmissão de dados é feita sem fio utilizando um XBee, permitindo que as informações sejam enviadas diretamente para o sistema central sem a necessidade de cabos, facilitando a instalação e a operação em áreas distantes. Além disso, o sistema possui um datalogger que armazena localmente os dados coletados em um cartão SD, garantindo que todas as medições estejam disponíveis para consulta posterior caso precise. A Estação de Monitoramento de Solo pode ser visualizada na Figura 6.

3.1.2 Controlador de Irrigação

O controlador de irrigação é responsável por acionar automaticamente o sistema de irrigação com base nas medições recebidas via rádio XBee das estações de monitoramento. Este módulo de acionamento opera de maneira autônoma e possui 4 modos de funcionamento: manual, timer, automático e smart, cada um com diferentes níveis de controle

[1] www.epsilon.ind.br

Figura 6 – Estação de monitoramento de umidade do solo em campo. Fonte: (AUTOMAÇÃO, 2021).

sobre o acionamento das bombas e válvulas de irrigação. Com 6 relés de estado sólido, sendo um destinado à bomba e cinco para as válvulas solenoides, o controlador permite um gerenciamento do fluxo de água em diferentes setores.

O sistema também oferece a flexibilidade de configuração por meio de botões e um display de cristal líquido, permitindo ajustes de parâmetros como os limites superiores e inferiores de umidade para cada setor. O módulo controlador de irrigação pode ser visualizado na Figura 7.

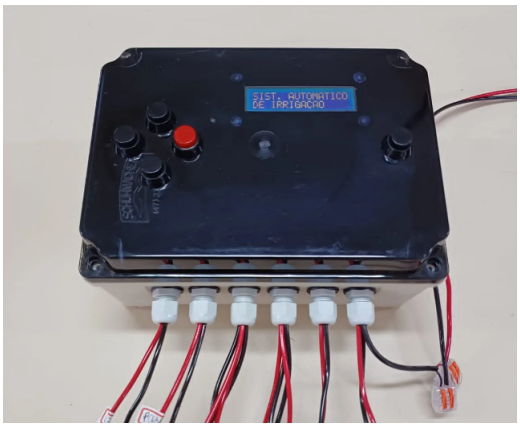

Figura 7 – Controlador de irrigação. Fonte: (AUTOMAÇÃO, 2021).

3.2 Descrição Geral do Sistema

O sistema de monitoramento e controle de irrigação abordado é composto por estações de medições de umidade do solo, que estão distribuídas em um campo agrícola.

Essas estações, ao coletarem cada medição, enviam as informações via comunicação sem fio para um controlador de irrigação como representado na Figura 8.

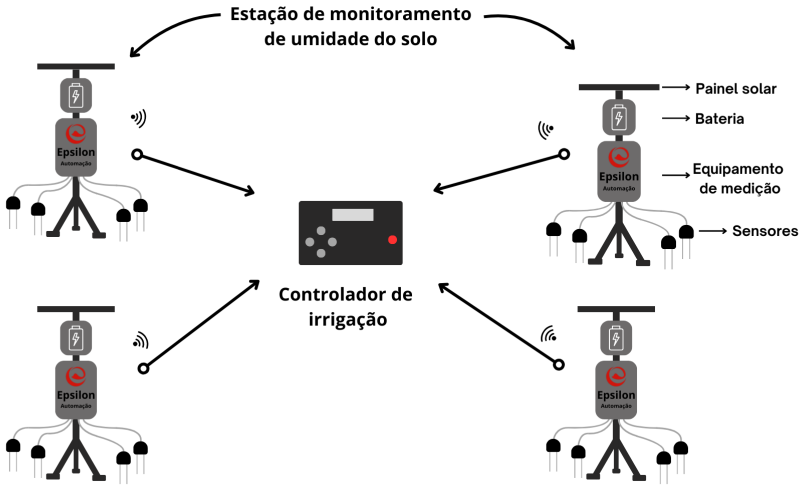

Figura 8 – Representação do sistema. Fonte: De autoria própria.

O controlador de irrigação, por sua vez, recebe essas medições e, dependendo da configuração de trabalho estabelecida, pode utilizar essas informações para tomar decisões automáticas sobre a irrigação do campo. Essas decisões podem incluir o acionamento ou desativação do sistema de irrigação em resposta aos níveis de umidade detectados, garantindo que o solo receba a quantidade ideal de água.

O sistema opera utilizando a topologia de rede DigiMesh do XBee, que permite que todos os dispositivos na rede se comportem como nós equivalentes. Isso significa que as estações de monitoramento enviam suas mensagens como nós da rede, e o controlador de irrigação também atua como um nó/roteador. Essa configuração garante que todas as mensagens sejam entregues de forma confiável, mesmo em ambientes onde a conectividade direta entre dispositivos pode não ser possível.

A topologia DigiMesh é vantajosa por eliminar a necessidade de um coordenador central, o que aumenta a robustez e a flexibilidade da rede. Todos os nós podem se comunicar entre si, formando uma malha onde as mensagens encontram automaticamente o melhor caminho até o destino. Isso contrasta com a arquitetura Zigbee tradicional, onde há a distinção entre coordenadores, roteadores e dispositivos finais, como ilustrado na Figura 9.

É importante destacar que, no estado atual sem a presença do *gateway*, o sistema opera de maneira totalmente offline. Ele foi inicialmente projetado para funcionar dessa forma, e a ausência de conectividade com a internet não impacta em nada a eficácia do sistema. O sistema foi desenvolvido para ser autossuficiente, realizando todas as operações e decisões localmente, sem a necessidade de comunicação com serviços externos.

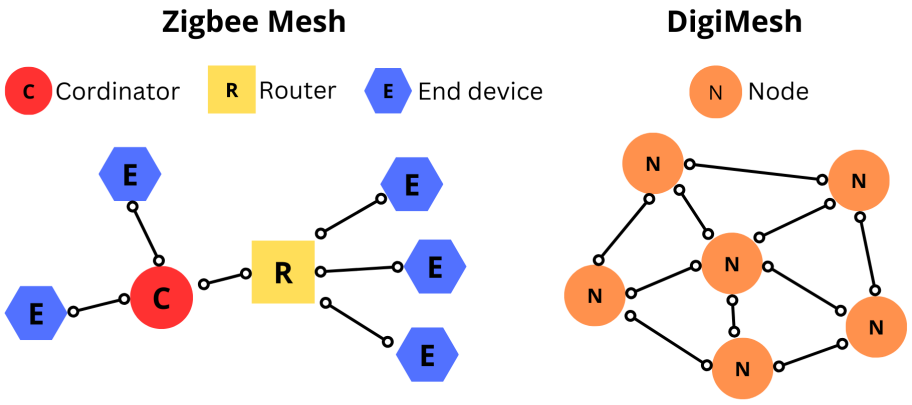

Figura 9 – Representação de protocolos de comunicação XBee. Fonte: De autoria própria.

3.3 Sistema conectado à internet

O *gateway* é então introduzido como um novo nó na rede, integrando-se à topologia DigiMesh do XBee. Ele é responsável por coletar todas as mensagens enviadas pelas estações de monitoramento de umidade do solo e pelo controlador de irrigação. Cada mensagem enviada na rede, seja por uma estação ou pelo controlador, é recebida pelo *gateway*.

É importante mencionar que o local onde o *gateway* está posicionado precisa ter conectividade Wi-Fi, garantindo que os dados coletados possam ser publicados em um *broker* MQTT para acesso remoto. Esse processo é detalhado mais adiante. A figura 10 a seguir ilustra como o *gateway* se integra ao sistema, recebendo as mensagens enviadas tanto pelas estações quanto pelo controlador.

3.4 Configuração do Hardware

O *gateway* é construído utilizando o SoC ESP8266, especificamente o WeMos D1 Mini, em conjunto com um módulo XBee para comunicação sem fio. Para facilitar a conexão entre o XBee e o WeMos, é utilizado um adaptador para XBee, que simplifica a integração dos dois dispositivos.

A Figura 11 ilustra as conexões físicas entre o WeMos D1 Mini e o adaptador XBee. O adaptador XBee é alimentado pelo pino Vcc (5V) do WeMos, enquanto o pino GND (terra) fornece o referencial de tensão. As conexões de comunicação são realizadas através dos pinos TX (D3) e RX (D4) do WeMos, que, neste caso, não fazem parte da UART (hardware serial) nativa do WeMos. Em vez disso, é utilizada uma serial simulada, criada

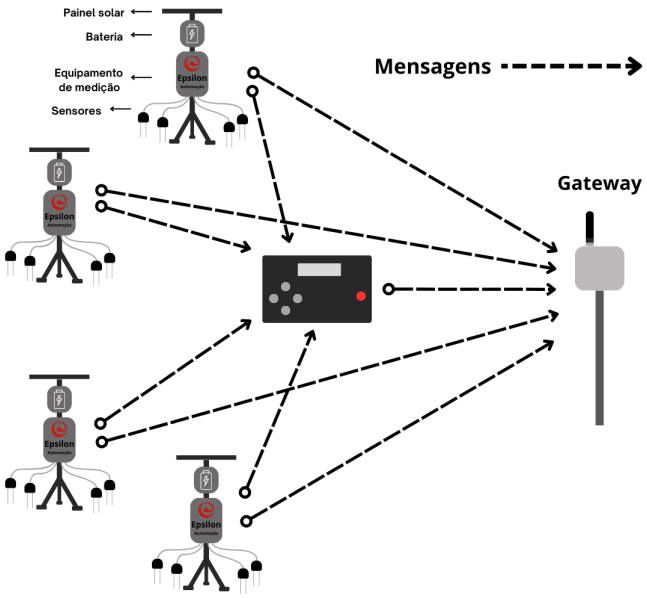

Figura 10 – Representação do sistema com a introdução do gateway. Fonte: De autoria própria.

com a biblioteca EspSoftwareSerial[2] .

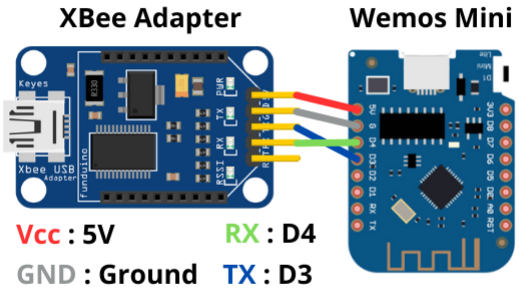

Figura 11 – Representação conexões físicas entre XBee e o Wemos. Fonte: De autoria própria.

O uso da EspSoftwareSerial permite maior flexibilidade no projeto, já que os pinos D3 e D4 estão convenientemente próximos à alimentação, facilitando a montagem. Além disso, ao não utilizar a serial nativa do hardware, é possível conectar o ESP8266 diretamente ao computador através da UART nativa, mantendo a capacidade de usar a serial para depuração e outras funcionalidades, sem interferir na comunicação com o módulo XBee.

[2] www.github.com/plerup/espsoftwareserial

Na Figura 12, é ilustrado o *gateway* fisicamente, o ESP8266 e o módulo XBee ficam acomodados dentro de uma caixa branca de proteção, com os cabos passando por um prensa-cabo, garantindo o máximo de isolamento. Esse cuidado é necessário, pois o *gateway* provavelmente será instalado em ambientes abertos pois precisa estar em um local que permita receber, sem obstáculos físicos, as mensagens enviadas pelas estações e pelo controlador de irrigação, fazendo que fique exposto às condições climáticas.

Figura 12 – Foto do gateway montado. Fonte: De autoria própria.

3.5 Software embarcado

3.5.1 Ambiente de Desenvolvimento e Linguagem de Programação

Durante o desenvolvimento do sistema embarcado, foram utilizadas diferentes ferramentas e ambientes para a programação do microcontrolador. Inicialmente, o Arduino IDE foi a plataforma escolhida devido à sua simplicidade e à vasta documentação disponível. O Arduino IDE é amplamente utilizado para projetos com microcontroladores da família ESP, como o ESP8266 e ESP32, sendo uma excelente opção para prototipagem rápida.

No entanto, à medida que o projeto avançou e se tornou mais complexo, optou-se pela migração para o PlatformIO. O PlatformIO é um ambiente de desenvolvimento integrado (IDE) que funciona como uma extensão do Visual Studio Code, oferecendo maior flexibilidade, recursos avançados de depuração e suporte a múltiplos ambientes de desenvolvimento. A integração do PlatformIO no Visual Studio Code permitiu uma melhor organização do projeto, maior controle sobre as bibliotecas utilizadas e uma facilidade na gestão de dependências e configurações específicas do hardware.

A linguagem de programação utilizada ao longo do projeto foi o C++, uma escolha comum em sistemas embarcados devido à sua eficiência e controle direto sobre os recursos do hardware. O C++ oferece uma abordagem estruturada e orientada a objetos, o que facilita a modularização do código e a reutilização de componentes em diferentes partes do projeto.

3.5.2 Funcionamento

Quando o *gateway* é iniciado, o primeiro passo é tentar se conectar à rede Wi-Fi utilizando as credenciais previamente armazenadas em sua memória não volátil (flash). Essas credenciais foram configuradas anteriormente, permitindo que o *gateway* se conecte automaticamente à rede sem a necessidade de intervenção do usuário.

Se o *gateway* não conseguir estabelecer a conexão com a rede Wi-Fi, ele entra em modo de configuração, gerando um Access Point (AP) próprio. Esse AP pode ser acessado por um dispositivo, como um celular, e permite que o usuário configure a rede Wi-Fi por meio de uma interface web representada pela Figura 13. Essa interface é fornecida por um servidor web (webserver) que é hospedado diretamente no ESP8266, utilizando a biblioteca WiFiManager. O WiFiManager facilita a configuração da rede, possibilitando a inserção de novas credenciais de Wi-Fi sem a necessidade de reprogramar o dispositivo.

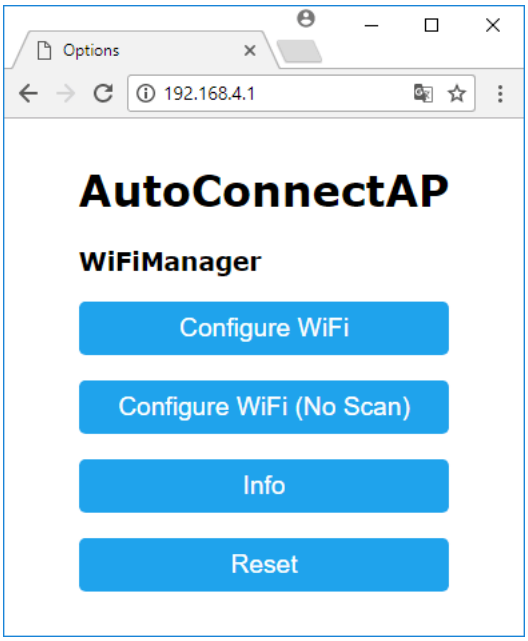

Figura 13 – Interace web hospedada pelo esp8266. Fonte: De autoria própria.

Uma vez que o *gateway* se conecta com sucesso à rede Wi-Fi, ele procede para configurar dois *brokers* MQTT. O primeiro é o *broker* de configuração, que é utilizado para receber comandos de configuração e ajustes operacionais do *gateway*. O segundo *broker* é responsável pela publicação das mensagens enviadas pelas estações de monitoramento

e pelo controlador de irrigação, para que os dados sejam transmitidos para a nuvem ou outro servidor remoto conforme necessário.

Após estabelecer a conexão com os *brokers* MQTT, o *gateway* entra em modo de espera, pronto para receber as mensagens provenientes das estações de monitoramento de umidade do solo e do controlador de irrigação.

3.6 Fluxo de mensagens

No sistema, as mensagens enviadas pelas estações de monitoramento de umidade do solo e pelo controlador de irrigação são estruturadas no formato de frames API do XBee. A Figura 14 se refere ao gerador de frames presente no Software XCTU muito utilizado na configuração dos módulos XBee, onde representa como um frame API é composto. Cada mensagem é iniciada com um delimitador (Start delimiter) e inclui campos como o comprimento da mensagem (Length), tipo de frame (Frame type), identificador do frame (Frame ID), endereço de destino de 16 bits(16-bit destination address), opções de transmissão (Options), mensagem em si (RF data), e um checksum para garantir a integridade da mensagem transmitida.

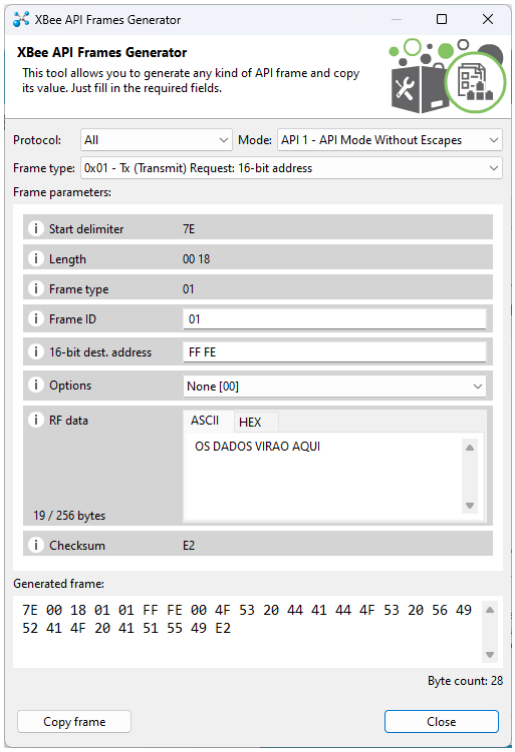

Figura 14 – Frame Generator. Fonte: Software XCTU.

A Figura 15 ilustra o fluxo de dados dentro do sistema, mostrando como as mensagens são transmitidas desde as estações de monitoramento e o controlador de

irrigação até o *gateway*. A imagem demonstra um exemplo de composição do frame API e o fluxo das mensagens enviadas e recebidas ao longo da rede.

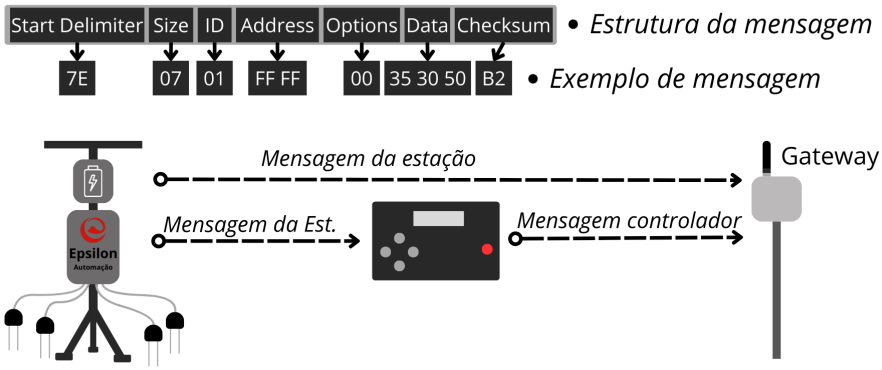

Figura 15 – Exemplo de troca de mensagem no sistema de irrigação Fonte: De autoria própria.

À medida que a mensagem é recebida pelo *gateway*, os bytes iniciais do frame API são verificados para assegurar que a estrutura da mensagem esteja correta. Em seguida, o byte final, que contém o checksum, é conferido para validar a integridade da mensagem. Caso todos os campos estejam corretos, a mensagem é então processada e formatada. A publicação no *broker* MQTT é realizada em um tópico específico, que é determinado pelo tamanho fixo da mensagem, correspondente a cada tipo de dado recebido. Essa abordagem garante que cada mensagem seja encaminhada para o tópico apropriado.

3.7 Integração com MQTT

Após a validação e formatação das mensagens recebidas, o *gateway* utiliza a biblioteca PubSubClient[3] para publicar os dados nos tópicos específicos do *broker* MQTT. A PubSubClient é uma biblioteca amplamente utilizada em dispositivos baseados em ESP8266 para gerenciar a comunicação com *brokers* MQTT. Ela oferece uma interface simples para conectar-se ao *broker*, publicar mensagens e subscrever-se a tópicos, permitindo que o ESP8266 funcione como um cliente MQTT completo.

A biblioteca PubSubClient gerencia automaticamente o envio das mensagens para o *broker* MQTT. Ao receber uma mensagem de um sensor ou do controlador de irrigação, o *gateway* a processa e, utilizando a PubSubClient, publica essa mensagem no tópico apropriado. Por exemplo, se a mensagem vem de um sensor de umidade do solo, ela é

[3] www.github.com/knolleary/pubsubclient

publicada no tópico Topico/Dados/Sensor. Já as mensagens enviadas pelo controlador são publicadas no tópico Topico/Dados/Controlador.

A Figura 16 ilustra como as mensagens são organizadas e publicadas nos diferentes tópicos do *broker*. Cada tópico é projetado para receber um tipo específico de mensagem, o que facilita o processamento e a distribuição das informações.

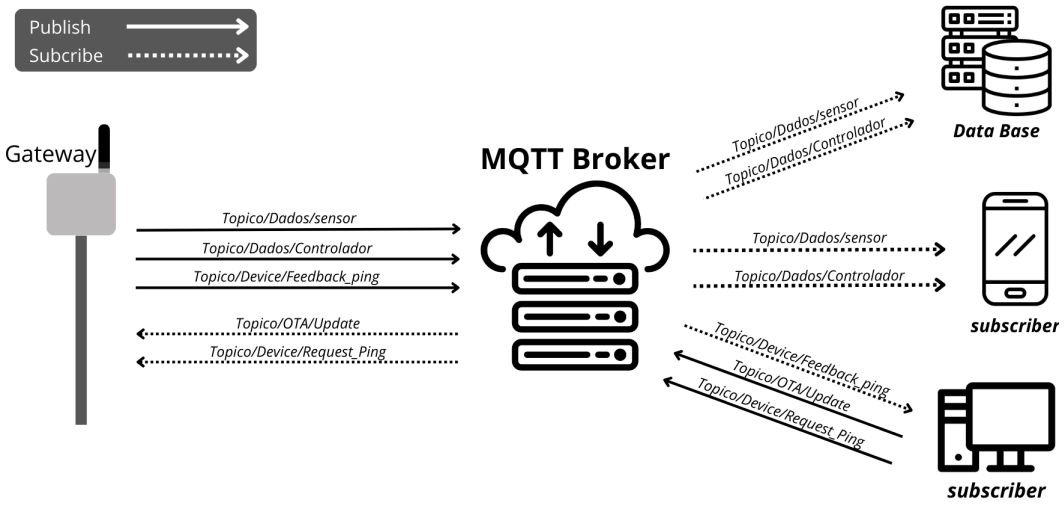

Figura 16 – Representação do fluxo de mensagens no MQTT. Fonte: De autoria própria.

Além da distribuição dos dados em tempo real, o sistema permite a integração com um banco de dados, onde as informações publicadas nos tópicos podem ser armazenadas para futuras consultas e análises. A PubSubClient também permite que o *gateway* publique em múltiplos tópicos ou realize assinaturas (subscriptions) em tempo real. Isso é especialmente útil para monitoramento histórico e para a tomada de decisões baseada em tendências observadas ao longo do tempo, especialmente em sistemas que demoram alterar seu estado.

O acesso aos dados do *broker* MQTT não se limita apenas a sistemas de back-end; os tópicos publicados podem ser acessados por dispositivos móveis, como celulares, ou por computadores, permitindo que os usuários monitorem o sistema em tempo real, de qualquer lugar. Aplicações específicas podem ser desenvolvidas para esses dispositivos, oferecendo uma interface para visualização e controle dos dados.

3.8 Atualização OTA

Foi implementada a funcionalidade de OTA para possibilitar a atualização remota do *firmware* dos dispositivos ESP8266. Cada dispositivo recebe um ID único durante a gravação inicial do *firmware*, o que permite identificar individualmente cada um ao longo do processo de atualização. O *firmware* básico gravado nesses dispositivos inclui apenas o

código responsável pelo processo de OTA e o gerenciamento de requisições para verificar por novas versões de *firmware*.

Para garantir a integridade e a segurança das atualizações, todos os dispositivos realizam requisições HTTPS periódicas para um repositório privado no GitHub, onde está hospedado um arquivo JSON (JavaScript Object Notation). Esse arquivo JSON contém uma lista com os IDs dos dispositivos que devem ser atualizados, além do endereço de onde o *firmware* atualizado pode ser baixado e a versão mais recente do código.

A Figura 17 ilustra o processo de requisição HTTPS dos dispositivos para o GitHub, onde cada dispositivo realiza uma requisição GET para verificar se há atualizações.

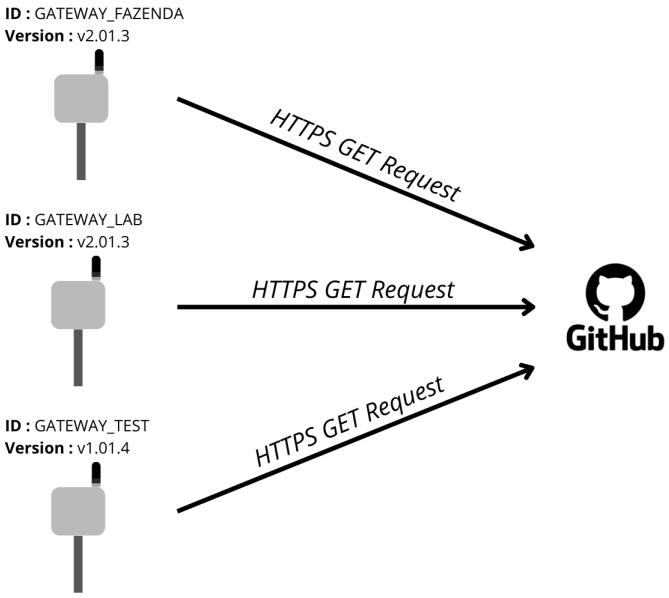

Figura 17 – Representação da requisição GET ao github Fonte: De autoria própria.

Quando o dispositivo faz a requisição HTTPS, ele compara a sua versão de *firmware* com a versão especificada no arquivo JSON. Se o dispositivo estiver na lista de atualização e a versão for diferente da atual, o novo *firmware* é baixado através de uma requisição GET para o endereço especificado no JSON. A Figura 18 mostra como o JSON é enviado como resposta, contendo as informações dos dispositivos que precisam de atualização.

A utilização do protocolo HTTPS é fundamental para garantir que as comunicações sejam criptografadas, protegendo o dispositivo contra ataques de interceptação e adulteração de dados durante o processo de atualização. Esse protocolo também garante que o *firmware* baixado seja autenticado e provenha de uma fonte confiável, uma vez que o ESP8266 valida os certificados de segurança durante a conexão.

Além da segurança proporcionada pelo HTTPS, o uso de repositórios privados do GitHub reforça o controle sobre quais dispositivos podem acessar os arquivos de *firmware*.

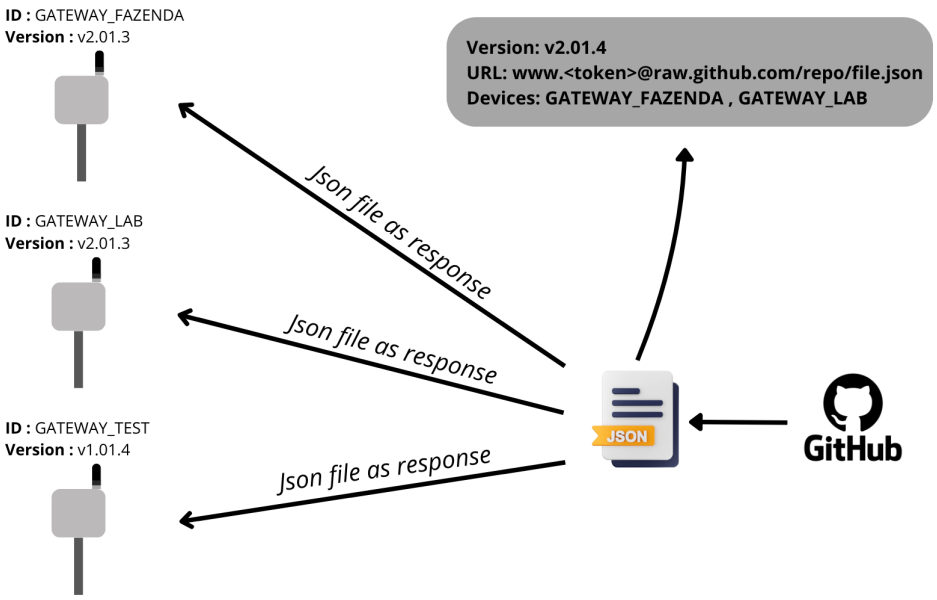

Figura 18 – Representação da resposta à requisição. Fonte: De autoria própria

O acesso aos repositórios é feito utilizando um token de autenticação, que garante que apenas dispositivos autorizados possam baixar o novo *firmware*. Essa abordagem protege o sistema contra tentativas não autorizadas de download ou modificação do *firmware*.

Uma vez que o arquivo binário (.bin) do novo *firmware* é baixado com sucesso, o dispositivo realiza a substituição do *firmware* anterior e executa um reset automático, aplicando a atualização sem a necessidade de intervenção manual. A Figura 19 demonstra o processo de resposta em que o arquivo .bin é baixado para os dispositivos selecionados.

Esse processo foi projetado para ser robusto, minimizando possíveis falhas durante a atualização e assegurando que os dispositivos estejam sempre em operação a versão mais recente do *firmware*, de acordo com as atualizações implementadas no repositório. O método de atualização OTA facilita a manutenção e o upgrade dos dispositivos em campo, reduzindo significativamente a necessidade de intervenções manuais e melhorando a escalabilidade do sistema de IoT.

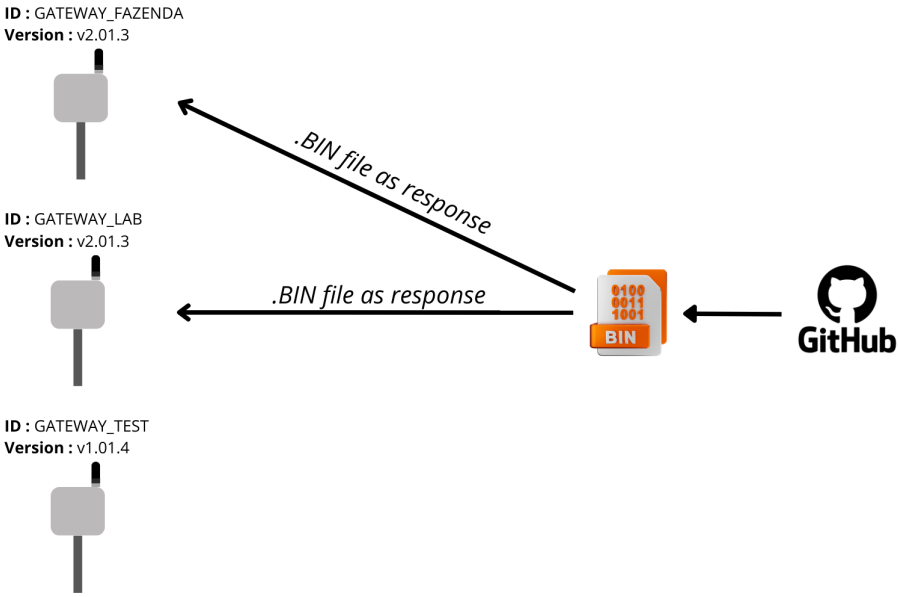

Figura 19 – Representação da resposta contendo o arquivo .bin. Fonte: De autoria própria.

4 Resultados

Neste capítulo, são apresentados os resultados dos testes realizados com o *gateway* em diferentes cenários. Primeiro, é mostrado o desempenho na plantação de algodão, onde o *gateway* recebeu dados apenas de uma estação de monitoramento de umidade do solo. Depois, é descrita a aplicação em uma plantação de verduras na UFOP, onde o sistema foi testado completo, com um controlador de irrigação e uma estação. Por fim, são discutidos os testes de atualização OTA, que verificaram a capacidade do *gateway* de atualizar remotamente os dispositivos conectados. Esses testes ajudaram a validar o funcionamento do sistema em situações práticas e variadas.

4.1 Testes em Aplicaçoes Reais

Além dos testes em laboratório realizados durante o desenvolvimento do projeto, o *gateway* foi colocado para funcionar em uma aplicação real. Para validar seu funcionamento em um ambiente prático, o *gateway* foi instalado em uma plantação de algodão localizada no oeste da Bahia, onde sua função principal era receber mensagens de uma estação.

A instalação foi realizada em um local estratégico, escolhido para garantir uma boa recepção das mensagens e minimizar possíveis obstáculos físicos que poderiam interferir na comunicação entre os XBee. O *gateway* foi posicionado em um local elevado, permitindo uma linha de visada mais clara entre ele e a estação. Além disso, o local escolhido também possuía acesso à internet, um fator importante para garantir que o *gateway* pudesse enviar e receber mensagens para a internet via MQTT.

4.2 Coleta e Armazenamento dos Dados pelo gateway

Os dados coletados pelo *gateway* entre 20/02/2024 e 01/07/2024 foram utilizados para ilustrar seu funcionamento. Durante esse período, o *gateway* foi responsável por receber as mensagens da estação de sensoriamento e enviá-las para um servidor remoto. Os dados foram transmitidos para um servidor rodando na AWS (Amazon Web Services), onde foram armazenados em um banco de dados. Esse processo garantiu que as informações enviadas pelo *gateway* estivessem acessíveis para consulta e análise.

4.2.1 Umidade Média Diária dos Sensores

A Figura 20 apresenta a umidade média diária coletada por cada sensor ao longo do período analisado. Esses dados foram enviados continuamente pelo *gateway* ao servidor,

demonstrando que o sistema foi capaz de receber as informações dos sensores de solo e transmitir para o banco de dados durante sua utilização.

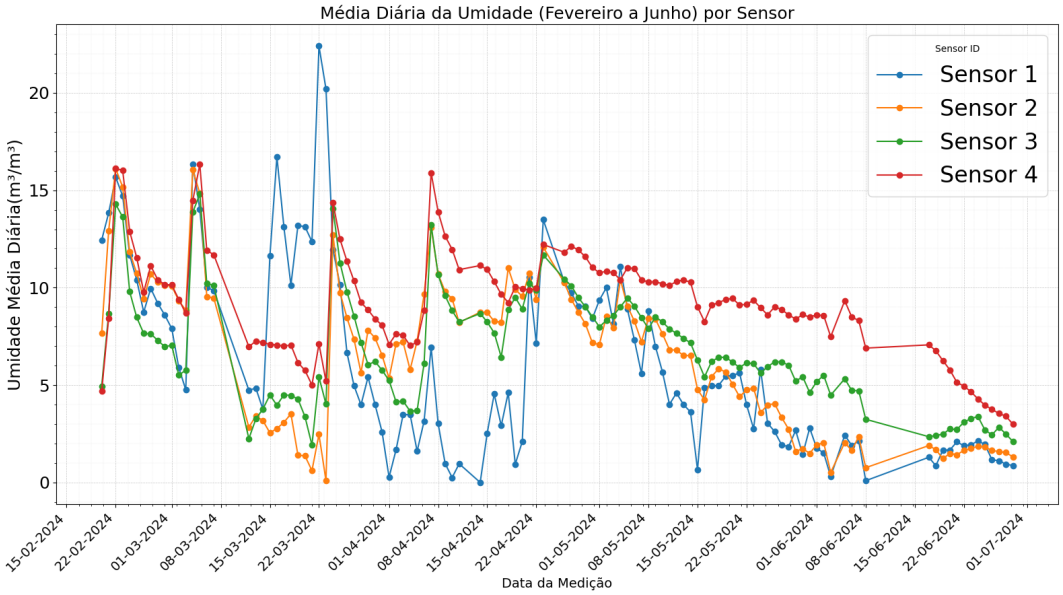

Figura 20 – Representação da média diária da umidade. Fonte: De autoria própria.

Variações significativas nos valores de umidade ao longo do tempo podem ser atribuídas principalmente aos ciclos de irrigação que ocorreram durante o período. Em determinados momentos, observam-se quedas nesses valores, o que pode indicar que o solo estava recebendo menos água, ou que houve uma pausa na irrigação. Além disso, descontinuidades em certos pontos dos dados podem ter ocorrido devido a interrupções no funcionamento do *gateway*, seja por falhas de conexão com a internet, manutenção no local, ou até mesmo desligamentos temporários para ajustes.

É importante notar que, ao final do período de coleta, os valores de umidade chegam a valores muito baixos para todos os sensores, o que coincide com o momento da colheita. Como não houve mais irrigação após essa etapa, a umidade naturalmente diminuiu, demonstrando o comportamento do sistema de monitoramento.

4.2.2 Tensão da Bateria e Temperatura dos Sensores e da Estação

A Figura 21 apresenta os dados de tensão da bateria e temperaturas dos sensores e da estação em um único dia (20/05/2024). Esse recorte foi selecionado para proporcionar uma visualização mais clara e detalhada das variações ocorridas ao longo do dia. Embora a imagem exiba apenas três variáveis, o pacote de dados enviado para o *gateway* contém diversos outros parâmetros, como a permissividade do solo, valores de variáveis armazenadas na memória do módulo, além de informações sobre o estado dos atuadores, configurações de modo de ativação, entre outros.

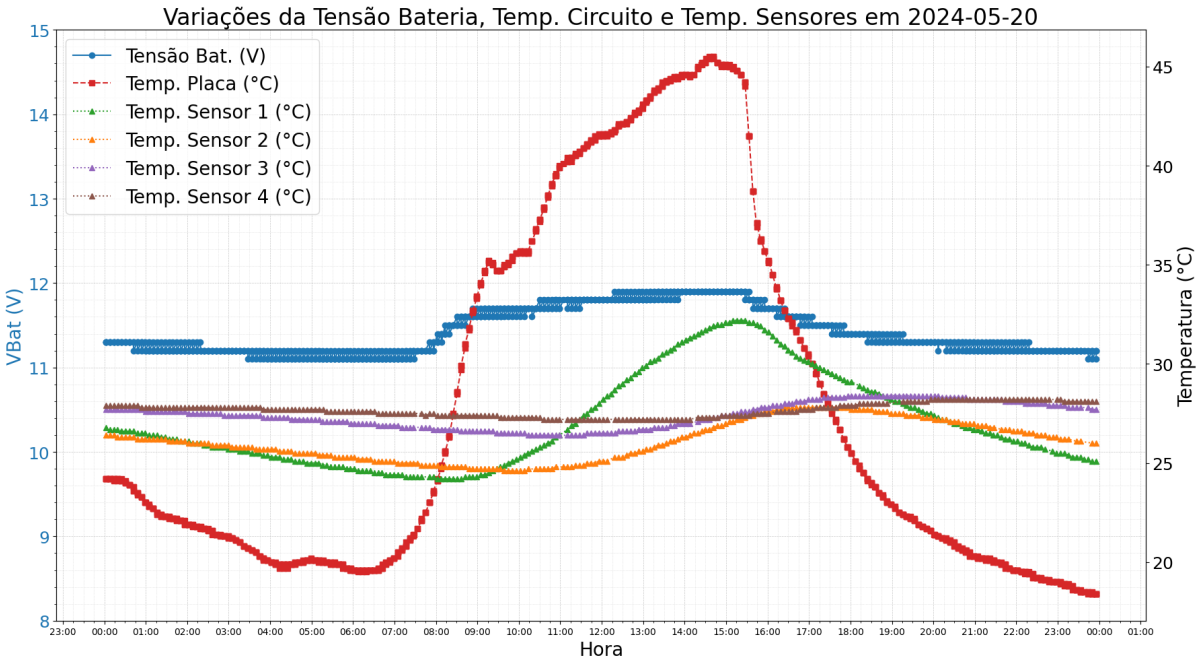

Figura 21 – Variação da Tensão da Bateria e Temperatura dos Sensores e da Estação no Dia 20/05/2024. Fonte: De autoria própria.

4.2.3 Análise da Perda de Pacote

Durante a análise da perda de pacotes (packet loss), foram observados comportamentos que, embora não impactem significativamente o monitoramento remoto ou as análises subsequentes, fornecem *insights* valiosos sobre o funcionamento do sistema e possíveis melhorias para o dispositivo. Casos em que nenhum dado foi recebido de nenhum dos quatro sensores foram desconsiderados na análise, uma vez que essas falhas podem ter sido causadas por problemas como quedas na conexão de internet ou interrupções no fornecimento de energia ao *gateway*.

A Figura 22 apresenta o número de mensagens recebidas e o total de dados perdidos para cada um dos sensores para o período de 20/02 até 01/07. Ao analisar os dados, é possível observar uma tendência de maior perda de pacotes de forma sequencial do sensor 1 ao 4, ou seja, o Sensor 2 apresenta uma maior taxa de perda em relação ao Sensor 1, o Sensor 3 perde mais pacotes do que o Sensor 2, e o Sensor 4 tem uma taxa de perda ainda maior em relação ao Sensor 3.

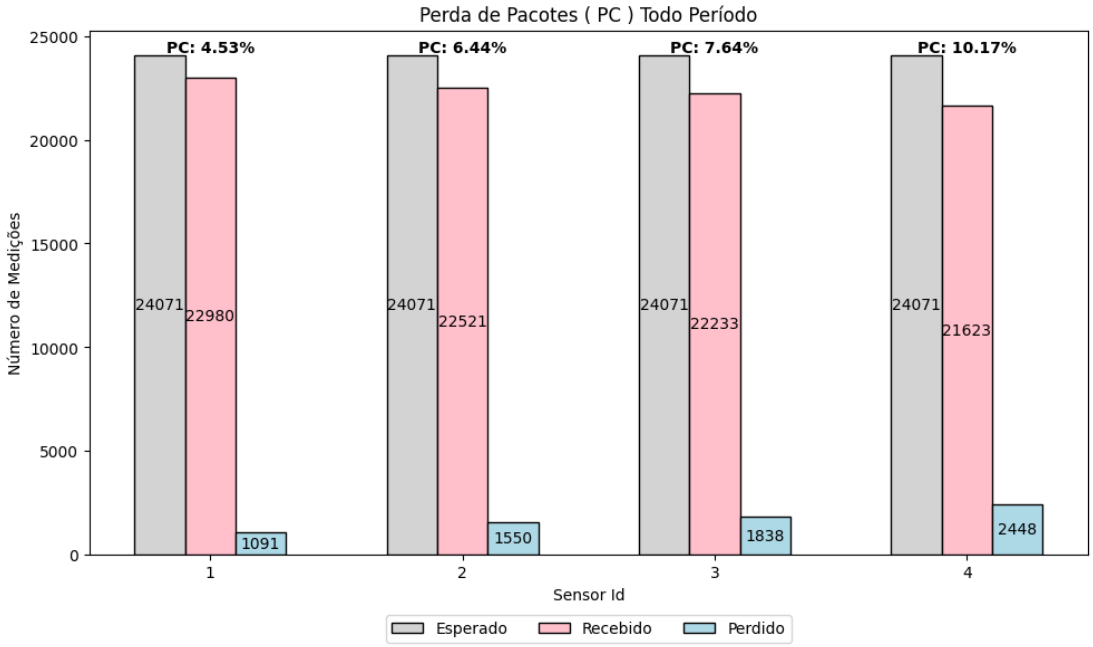

Figura 22 – Perda de Pacote Para Todo o Período. Fonte: De autoria própria.

Algumas hipóteses podem ser levantadas para explicar essa diferença na perda de pacotes entre os sensores. Como as mensagens são enviadas de forma sequencial pelas estações de monitoramento, é possível que, ao tentar enviar quatro mensagens seguidas para o broker MQTT, o tempo entre as tentativas de publicação esteja sendo muito curto. Isso poderia causar sobrecarga na transmissão dos dados, especialmente nas mensagens que contêm os dados dos sensores 3 e 4, que apresentam taxas de perda mais elevadas.

Embora não tenha sido realizado um experimento específico para identificar a origem exata das perdas, essas observações indicam que os problemas podem estar relacionados à velocidade e ao intervalo entre as transmissões dos pacotes. Uma investigação mais aprofundada, com experimentos adicionais, pode ser necessária para confirmar essas suposições e para ajustar o *gateway*.

Ao examinar a Figura 23 sobre a quantidade de mensagens perdidas consecutivas por sensor, fica claro que a maior parte das perdas corresponde à perda de apenas uma mensagem consecutiva. Esse comportamento é observado em todos os 4 sensores, o que significa que, mesmo em períodos com perdas, estas ocorrem de forma isolada, sem que uma sequência de múltiplos pacotes seja perdida.

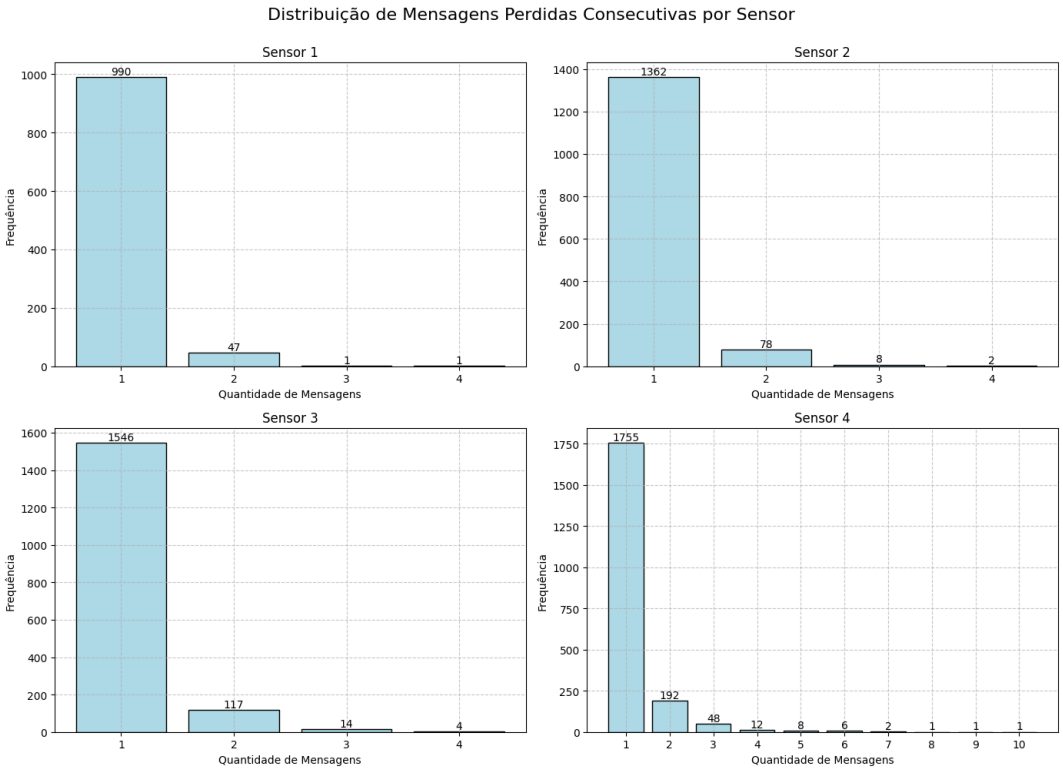

Figura 23 – Perda de Pacote Consecutivos. Fonte: De autoria própria.

Portanto, esses episódios de perda pontual de pacotes não comprometem a análise remota dos dados, uma vez que a perda de um único pacote não impacta de maneira significativa a média diária de qualquer variável monitorada, garantindo assim, a confiabilidade das medições.

Foi realizada também uma análise da média diária de perda de dados consecutivos, apresentada no gráfico representado pela Figura 24. Essa análise mostra que, apesar das perdas pontuais de pacotes, a média diária de dados perdidos se mantém relativamente baixa para todos os sensores. Além disso, foi calculado o desvio padrão, que ajuda a entender a dispersão das perdas em relação à média.

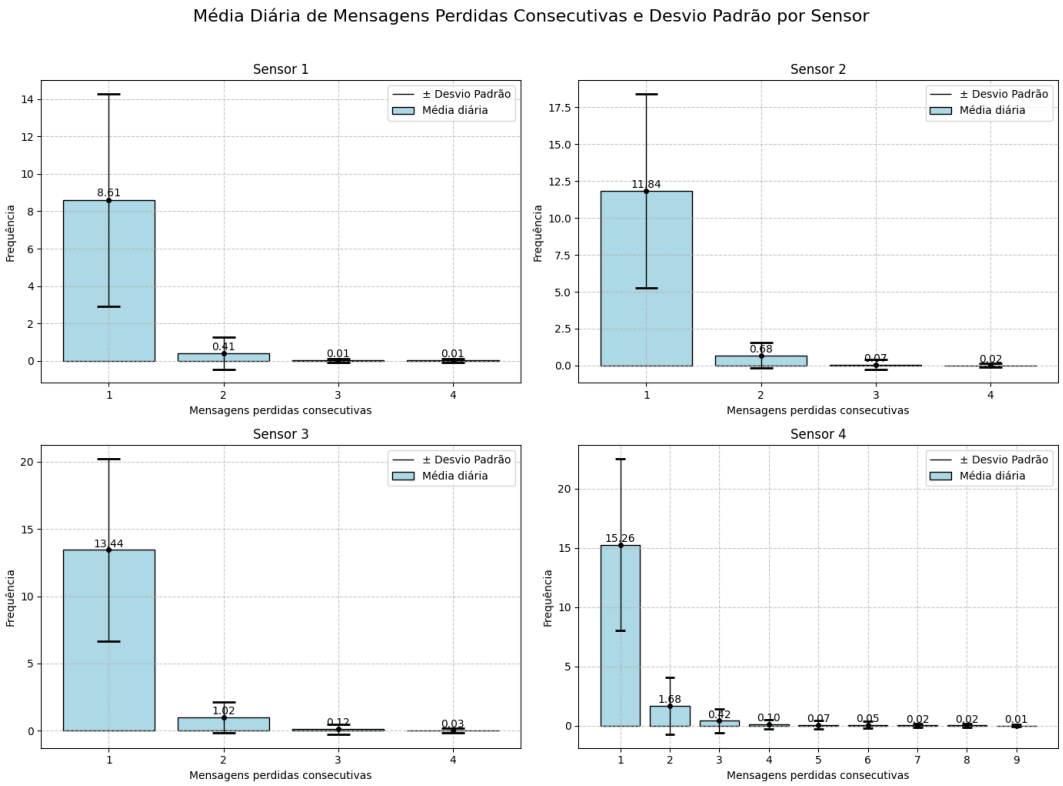

Figura 24 – Média da Perda de Pacote Consecutivos e Desvio Padrão. Fonte: De autoria própria.

Por fim, foi feito um recorte de um período de 14 dias e gerado um gráfico representado pela Figura 25 abrangendo o intervalo de 01/05/2024 a 13/05/2024. Nele, a linha vermelha que representa o *packet loss* ao longo dos dias permite uma visualização de como a perda de dados ocorre durante esse período. A quantidade de medições realizadas é de aproximadamente 260 por dia por sensor devido o tempo de amostragem configurado na estação de monitoramento. A linha exibe a variação diária na taxa de perda de pacotes, destacando os dias com maiores ou menores perdas, o que ajuda a identificar possíveis padrões ou períodos de maior instabilidade no sistema.

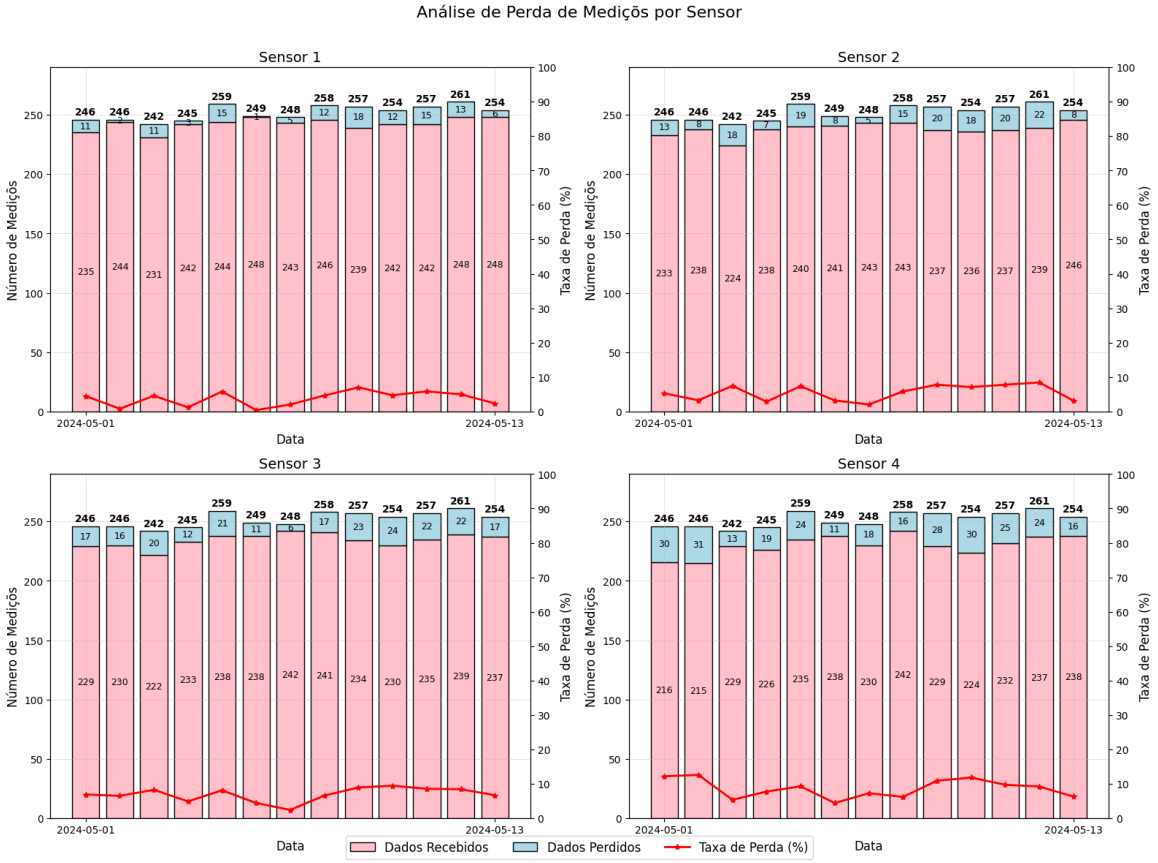

Figura 25 – Perda de Pacote Durante 2 Semanas. Fonte: De autoria própria.

4.3 Aplicação em Plantação de Verduras na UFOP

Como parte dos testes de validação do sistema desenvolvido, o *gateway* também foi aplicado em uma pequena plantação de verduras situada no canteiro entre os prédios que abrigam os laboratórios da Escola de Minas da UFOP. Esse cenário foi fundamental, pois tem permitido ajustes e melhorias no comportamento dos dispositivos constantemente.

O sistema nessa aplicação é composto por uma estação de monitoramento equipada com quatro sensores de umidade do solo, um controlador de irrigação responsável por irrigar um único setor, e o *gateway*, que realiza a comunicação e coleta de dados da estação e do controlador. A estação de monitoramento instalada na plantação é representada na Figura 26.

O controlador de irrigação, ilustrado na Figura 27, foi configurado para gerir apenas um setor de irrigação, simulando uma aplicação especifica para validação do comportamento do sistema. Ele se comunica com o *gateway* por meio do XBee e ainda permite a atuação direta sobre a irrigação com base nos dados recebidos da estação.

Já o gateway, representado na Figura 28, foi instalado dentro do laboratório LABCAM, garantindo acesso contínuo à internet, permitindo a coleta e envio dos dados

Figura 26 – Estação de monitoramento presente na UFOP. Fonte: De autoria própria.

Figura 27 – Controlador de irrigação presente na UFOP. Fonte: De autoria própria.

ao servidor remoto e também para a aplicação de atualizações OTA.

4.3.1 Medições de Umidade

Como uma possível formato de analise dos dados coletados, o gráfico representado pela Figura 29 mostra as medições de umidade do solo realizadas entre os dias 27 de setembro e 1 de outubro de 2024, coletadas pelos quatro sensores presentes na plantação

Figura 28 – *gateway* instalado no laboratório. Fonte: De autoria própria.

de verduras. Cada curva representa o comportamento da umidade em uma área específica monitorada por cada um deles. Os picos de umidade, bem visíveis em cada curva, indicam os momentos de irrigação, seguidos pela diminuição progressiva dos níveis de umidade à medida que a água é absorvida pelo solo e ocorre evaporação ao longo do dia.

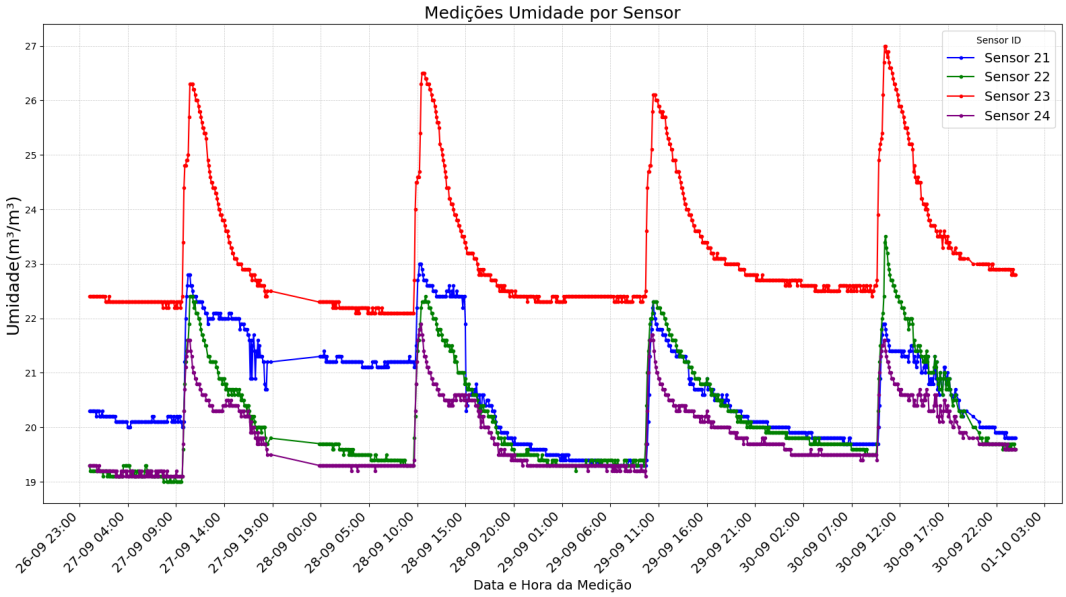

Figura 29 – Exemplo de análise dos dados coletados da estação. Fonte: De autoria própria.

Após cada irrigação, a queda gradual da umidade medida por todos os sensores reflete a dinâmica natural do solo, permitindo observar como o solo retém e perde água ao longo do tempo. Essa variação é uma possível forma de avaliar se o tempo de irrigação

foi suficiente para atender às necessidades das plantas e se os ciclos de irrigação estão adequados para manter o solo nas condições certas.

Além disso, é possível notar diferenças na curva de cada sensor, o que pode indicar pequenas variações na capacidade de retenção de água entre diferentes pontos da plantação, possivelmente devido a diferenças na composição ou compactação do solo. Essas variações também podem ser resultado de uma instalação inadequada dos sensores, como uma profundidade muito superficial ou um posicionamento incorreto. A possibilidade de analisar esses dados em tempo real é extremamente útil, pois permite identificar problemas com os equipamentos de monitoramento, como sensores defeituosos ou mal instalados, além de sugerir melhorias no sistema de irrigação para otimizar e garantir a distribuição de água correta.

4.3.2 Status da Bomba e Tempo Restante de Irrigação

O segundo gráfico representado pela Figura 30 é mais um exemplo de como os dados armazenados no banco de dados podem ser analisados. O status da bomba de irrigação, representado pela linha azul, e o tempo restante de irrigação, representado pela linha laranja, durante um breve período de operação. A linha azul indica os momentos em que a bomba foi ativada e desativada, alternando entre os estados ativa (1) e inativa (0). Conforme as configurações do controlador, a bomba é programada para irrigar durante 50 minutos por dia, o que pode ser observado na redução gradativa do tempo restante de irrigação ao longo da sessão, representado pela linha laranja.

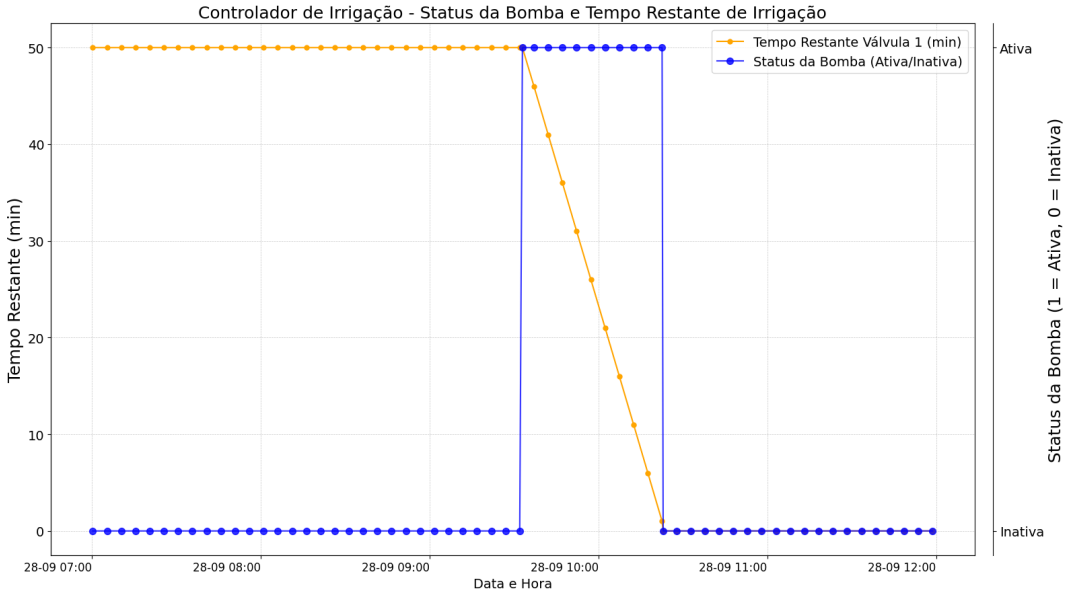

Figura 30 – Exemplo de análise dos dados coletados do controlador de irrigação. Fonte: De autoria própria.

Esse gráfico revela o comportamento programado do sistema, mostrando o acio-

namento e desativação da bomba, além de indicar o tempo de cada ciclo de irrigação. O intervalo constante entre os momentos de ativação e desativação reflete o cumprimento das configurações predefinidas no controlador, validando o desempenho do equipamento em operação.

Os dados mostrados neste gráfico são apenas uma parte das várias informações que são enviadas e armazenadas no banco de dados. Além do status da bomba e do tempo de irrigação, a mensagem salva no banco de dados também contém outros dados relevantes, como o tempo de irrigação para cada setor, qual setor está ativo no momento e qual o modo de operação em que o controlador está configurado.

4.4 Testes de Atualização OTA

Para validar o sistema de atualização OTA implementado no projeto, foram realizados testes práticos com dois dispositivos ESP8266. O processo de atualização foi acompanhado através de logs detalhados impressos na interface serial, o que permitiu visualizar todas as etapas envolvidas na verificação e aplicação da atualização. A Figura 31 apresenta os logs gerados durante o processo de um dos dispositivos, ESP_TEST_OTA_1.

```
1 [CONF_MQTT] Inicializando dispositivo de ID: ESP_TEST_OTA_1
2 [CONF_MQTT] Versao do firmware: v1.02.6
3
4 [CONF_MQTT] Conferindo versao do firmware... * Versao atual: v1.02.6   * Versao disponivel: v1.02.6 - Sem nova versao disponivel.
5 [CONF_MQTT] Conferindo versao do firmware... * Versao atual: v1.02.6   * Versao disponivel: v1.02.6 - Sem nova versao disponivel.
6 [CONF_MQTT] Conferindo versao do firmware... * Versao atual: v1.02.6   * Versao disponivel: v1.02.7 - Nova versao disponivel.
7
8 [CONF_MQTT] Comecando OTA... Preparando dispositivo para atualizacao... Baixando conteudo do github...
9
10 [httpUpdate] Process - Server header: - code: 200     - len: 513952
11 [httpUpdate] ESP8266 info: - free Space: 2629632     - current Sketch Size: 513904
12
13 //########################################//
14 //          DISPOSITIVO REINICIALIZADO      //
15 //########################################//
16
17 [CONF_MQTT] Inicializando dispositivo de ID: ESP_TEST_OTA_1
18 [CONF_MQTT] Versao do firmware: v1.02.7
19
20 [CONF_MQTT] Conferindo versao do firmware... * Versao atual: v1.02.7   * Versao disponivel: v1.02.7 - Sem nova versao disponivel.
21 [CONF_MQTT] Conferindo versao do firmware... * Versao atual: v1.02.7   * Versao disponivel: v1.02.7 - Sem nova versao disponivel.
22
```

Figura 31 – Logs gerados na atualização OTA. Fonte: De autoria própria.

Os dois dispositivos estavam inicialmente na versão de *firmware* v1.02.6. Cada dispositivo realiza verificações periódicas em um arquivo JSON armazenado no GitHub, conferindo se uma nova versão do *firmware* está disponível. No início do teste, os logs indicavam que não havia atualização, uma vez que a versão disponível era a mesma que os dispositivos já possuíam. No entanto, após a atualização do arquivo JSON no repositório do GitHub, o dispositivo detectou que a nova versão v1.02.7 estava disponível.

A partir desse momento, ambos os dispositivos iniciaram o processo de atualização OTA, que envolveu o download do novo *firmware* diretamente do GitHub via HTTPS. Após o download, os dispositivos foram reinicializados automaticamente, conforme demonstrado

nos logs. Ao reiniciar, as verificações confirmaram que o *firmware* havia sido atualizado com sucesso para a versão v1.02.7. Uma foto do experimento pode ser vista na Figura 32.

Figura 32 – Foto dos Wemos conectados à porta COM do computador. Fonte: De autoria própria.

Os logs dos dois dispositivos foram praticamente idênticos, diferindo apenas no nome do dispositivo. Esse teste poderia ser facilmente expandido para qualquer número de dispositivos, pois o sistema de OTA do *gateway* foi projetado para atualizar qualquer dispositivo cujo ID esteja listado no arquivo JSON.

5 Conclusão

Este trabalho teve como objetivo o desenvolvimento e implementação de um *gateway* utilizando o SoC ESP8266 e o protocolo MQTT, visando o monitoramento remoto de um sistema de irrigação. Durante o desenvolvimento, o *gateway* foi projetado para coletar dados de estações de monitoramento de umidade do solo e de controladores de irrigação, além de permitir atualizações remotas de *firmware* por meio do processo de OTA. Os testes realizados confirmaram a eficiência do sistema e a capacidade do *gateway* de integrar os dispositivos com a nuvem, garantindo um desempenho confiável.

A análise da eficiência do *gateway* na recepção e retransmissão de mensagens demonstrou que o protocolo MQTT foi eficaz na comunicação entre o *gateway* e o servidor, sem comprometer a confiabilidade dos dados. As perdas de pacotes foram mínimas, e não impactaram de forma significativa as medições diárias, como evidenciado pelos testes realizados durante o período de análise.

A avaliação da confiabilidade do sistema de atualização remota do *firmware* mostrou que o processo de OTA se revelou robusto, com sucesso na atualização do dispositivo sem falhas notáveis. A realização de testes em laboratório comprovou a estabilidade do sistema, e as atualizações ocorreram sem interrupções ou comprometimento do desempenho geral do *gateway*. A possibilidade de realizar atualizações de *firmware* através do OTA torna o *gateway* uma solução escalável a longo prazo, permitindo que melhorias contínuas sejam implementadas sem a necessidade de manutenção manual no local.

Além disso, o estudo revelou que, embora o sistema tenha apresentado um baixo índice de perda de pacotes, é possível que eventuais falhas na comunicação possam ser mais bem compreendidas por meio da implementação de mecanismos para diagnosticar as causas dessas perdas, como problemas de interferência, falhas de rede ou limitações no desempenho do hardware. Essas melhorias permitiriam uma análise mais detalhada e assertiva do funcionamento do sistema, possibilitando ajustes mais precisos.

No futuro, o sistema poderá ser aprimorado com a implementação de atualizações OTA também para os dispositivos conectados, como a estação de monitoramento e o controlador de irrigação, o que eliminaria ainda mais a necessidade de intervenções físicas, trazendo maior autonomia e agilidade ao sistema como um todo. Tais melhorias contribuirão para a evolução do sistema, aumentando sua eficiência e a capacidade de operar de maneira ainda mais autônoma e confiável.

Referências

AUTOMAÇÃO, Epsilon. *epsilon*. 2021. Disponível em: https://www.epsilon.ind.br. Acesso em: 12 set. 2024. Citado 2 vezes na página 23.

AXELSON, Jan. *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*. 2nd: Lakeview Research LLC, 2019. Citado 4 vezes nas páginas 19, 20.

COMMUNITY, Arduino. *OTA Updates on Arduino ESP8266*. 2024. Accessed: 2024-09-29. Disponível em: www.arduino-esp8266.readthedocs.io/en/latest/ota_updates/readme.html. Citado 1 vez na página 17.

DIGI INTERNATIONAL. *XBee/XBee-PRO RF Modules - 802.15.4*. Set. 2009. Data sheet do módulo XBee/XBee-PRO RF - 802.15.4. Citado 1 vez na página 19.

DIGI INTERNATIONAL. *Explore the Digi XBee Ecosystem*. Disponível em: https://www.digi.com/xbee. Acesso em: 20 ago. 2024. Citado 2 vezes na página 18.

DIGI INTERNATIONAL. *XBee/XBee-PRO S1 802.15.4 (Legacy) User Guide*. 2018. Revision W, June 2017. Disponível em: https://www.digi.com. Citado 2 vezes na página 19.

DIGI INTERNATIONAL. *XBee/XBee-PRO ZB RF Modules User Guide*. 2022. Revision AB, March 2022. Disponível em: https://www.digi.com. Citado 1 vez na página 19.

ELIJAH, Olakunle et al. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet of Things Journal*, v. 5, n. 5, p. 3758–3773, 2018. DOI: 10.1109/JIOT.2018.2844296. Citado 2 vezes na página 9.

ESPRESSIF SYSTEMS. *ESP-IDF OTA Documentation*. 2024. Accessed: 2024-09-29. Disponível em: www.docs.espressif.com/projects/esp-idf/en/release-v3.0/api-reference/system/ota.html. Citado 2 vezes na página 17.

ESPRESSIF SYSTEMS. *ESP8266 Technical Reference*. 2020. Version 1.7. Disponível em: https://www.espressif.com/en/company/documents/documentation_feedback?docId=1690&sections=&version=1.6. Citado 4 vezes na página 16.

FROIZ-MÍGUEZ, Iván et al. Design, Implementation, and Empirical Validation of an IoT Smart Irrigation System for Fog Computing Applications Based on LoRa and LoRaWAN Sensor Nodes. *Sensors*, v. 20, n. 23, 2020. ISSN 1424-8220. DOI: 10.3390/s20236865. Disponível em: https://www.mdpi.com/1424-8220/20/23/6865. Citado 1 vez na página 20.

GAST, Matthew. *802.11 Wireless Networks: The Definitive Guide*. 2nd: O'Reilly Media, 2005. ISBN 978-0596100520. Citado 1 vez na página 15.

GLÓRIA, André; CERCAS, Francisco; SOUTO, Nuno. Design and implementation of an IoT gateway to create smart environments. *Procedia Computer Science*, v. 109, p. 568–575, 2017. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.343. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050917310128. Citado 1 vez na página 9.

GOURLEY, David; TOTTY, Brian. *HTTP: The Definitive Guide*. Sebastopol, CA: O'Reilly Media, 2002. 1st edition. Citado 2 vezes na página 12.

GREENGARD, Samuel. *The Internet of Things*. The MIT Press, 2015. (Essential Knowledge Series). ISBN 9780262328920. Disponível em: http://gen.lib.rus.ec/book/index.php?md5=06e78474d7f59ef9a16b6bcb5f743f62. Citado 1 vez na página 9.

HABIB, Shabana et al. Design and Implementation: An IoT-Framework-Based Automated Wastewater Irrigation System. *Electronics*, v. 12, n. 1, 2023. ISSN 2079-9292. DOI: 10.3390/electronics12010028. Disponível em: https://www.mdpi.com/2079-9292/12/1/28. Citado 1 vez na página 21.

HILLAR, Gaston C. *MQTT Essentials - A Lightweight IoT Protocol*. Packt Publishing, 2017. ISBN 1787287815,9781787287815. Disponível em: http://gen.lib.rus.ec/book/index.php?md5=69a841763c1c75ef46d7a4b1f7285e54. Citado 1 vez na página 14.

HOROWITZ, Paul; HILL, Winfield. *The Art of Electronics*. 3rd: Cambridge University Press, 2015. Citado 2 vezes nas páginas 19, 20.

KILARU, Aditya Sai et al. Automatic Remote Farm Irrigation System with WSN and Weather Forecasting. *Journal of Physics: Conference Series*, IOP Publishing, v. 2161, n. 1, p. 012075, jan. 2022. DOI: 10.1088/1742-6596/2161/1/012075. Disponível em: https://dx.doi.org/10.1088/1742-6596/2161/1/012075. Citado 1 vez na página 20.

KOLBAN, Neil. *Kolban's Book on ESP32*. 1st: Leanpub, 2017. Citado 2 vezes nas páginas 15, 16.

LOU, Xiaokang et al. Design of Intelligent Farmland Environment Monitoring System Based on Wireless Sensor Network. *Journal of Physics: Conference Series*, IOP Publishing, v. 1635, n. 1, p. 012031, nov. 2020. DOI: 10.1088/1742-6596/1635/1/012031. Disponível em: https://dx.doi.org/10.1088/1742-6596/1635/1/012031. Citado 1 vez na página 20.

MISHRA, Biswajeeban; KERTESZ, Attila. The Use of MQTT in M2M and IoT Systems: A Survey. *IEEE Access*, v. 8, p. 201071–201086, 2020. DOI: 10.1109/ACCESS.2020.3035849. Citado 2 vezes nas páginas 13, 14.

MURALI, Kola; SRIDHAR, B. A smart Agriculture Irrigation System using sensor array based IOT. *Journal of Physics: Conference Series*, IOP Publishing, v. 2062, n. 1, p. 012010, nov. 2021. DOI: 10.1088/1742-6596/2062/1/012010. Disponível em: https://dx.doi.org/10.1088/1742-6596/2062/1/012010. Citado 1 vez na página 20.

QUY, Vu Khanh et al. IoT-Enabled Smart Agriculture: Architecture, Applications, and Challenges. *Applied Sciences*, v. 12, n. 7, 2022. ISSN 2076-3417. DOI: 10.3390/app12073396. Disponível em: https://www.mdpi.com/2076-3417/12/7/3396. Citado 1 vez na página 9.

SCHWARTZ, Marco. *Internet of Things with ESP8266: Build amazing IoT projects using the ESP8266 Wi-Fi chip*. Packt Publishing, 2016. ISBN 978-1786468024. Citado 1 vez na página 15.

SHKLAR, Leon; ROSEN, Richard. *Web Application Architecture: Principles, Protocols and Practices*. Chichester, England: John Wiley & Sons Ltd, 2003. 1st edition. Citado 2 vezes nas páginas 12, 13.

TANENBAUM, Andrew S. *Computer Networks*. 5th: Prentice Hall, 2011. ISBN 978-0132126953. Citado 2 vezes na página 15.

VALDES-PEREZ, Fernando E.; HERNANDEZ-AVELAR, Rogelio D. *Microcontrollers: Fundamentals and Applications with PIC*. Springer, 2009. ISBN 978-1441906049. Citado 1 vez na página 14.

VALVANO, Jonathan. *Embedded Systems: Real-Time Operating Systems for ARM Cortex M Microcontrollers*. CreateSpace Independent Publishing Platform, 2012. ISBN 978-1477508992. Citado 1 vez na página 14.

WARDANA, I Nyoman Kusuma et al. IoT-based Drip Irrigation Monitoring and Controlling System using NodeMCU and Raspberry Pi. *REVISTA DE TECNOLOGIA APLICADA*, Atlantis Press, p. 557–560, 2018/12. ISSN 2589-4943. DOI: 10.2991/icst-18.2018.115. Disponível em: https://doi.org/10.2991/icst-18.2018.115. Citado 1 vez na página 21.

WILMSHURST, Tim. *Designing Embedded Systems with PIC Microcontrollers: Principles and Applications*. 2nd: Newnes, 2009. ISBN 978-1856177504. Citado 1 vez na página 14.

XU, Jinyuan; GU, Baoxing; TIAN, Guangzhao. Review of agricultural IoT technology. *Artificial Intelligence in Agriculture*, v. 6, p. 10–22, 2022. ISSN 2589-7217. DOI: https://doi.org/10.1016/j.aiia.2022.01.001. Disponível em: https://www.sciencedirect.com/science/article/pii/S2589721722000010. Citado 1 vez na página 9.