UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

SANTIAGO COELHO TEIXEIRA

# PROJETO DE IRRIGAÇÃO AUTOMÁTICA COM MONITORAMENTO REMOTO E *ENERGY HARVESTING*

Sorocaba
2025

SANTIAGO COELHO TEIXEIRA

# PROJETO DE IRRIGAÇÃO AUTOMÁTICA COM MONITORAMENTO REMOTO E *ENERGY HARVESTING*

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba, como parte dos requisitos para obtenção do grau de Bacharel(a) em Engenharia de Controle e Automação.

Orientador(a): Prof. Dr. Eduardo Verri Liberado

Sorocaba

2025

T266p

Teixeira, Santiago Coelho
Projeto de irrigação automática com monitoramento remoto e energy harvesting / Santiago Coelho Teixeira. -- Sorocaba, 2025
67 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Eduardo Verri Liberado

1. Automação. 2. Sensoriamento remoto. 3. Internet das coisas. 4. Colheita de energia. 5. Aplicativos móveis. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

SANTIAGO COELHO TEIXEIRA

# PROJETO DE IRRIGAÇÃO AUTOMÁTICA COM MONITORAMENTO REMOTO E *ENERGY HARVESTING*

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel(a) em Engenharia de Controle e Automação.

Data da defesa: 12/03/2025

**BANCA EXAMINADORA:**

Prof. Dr. Eduardo Verri Liberado
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Prof. Dr. Leopoldo André Dutra Lusquino Filho
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Prof. Dr. Luis De Oro Arenas
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

## AGRADECIMENTOS

Aos meus pais, Wilson Roberto e Querzia, pela educação e base familiar que me permitiram chegar até aqui, além das sugestões sobre o projeto, já que serão os usuários finais.

Ao meu irmão Felipe, pela compra e instalação dos dutos de irrigação no jardim de inverno, onde realizei os testes do projeto.

À minha namorada, Rafaela Carolina, pelo apoio durante a faculdade e pelas sugestões sobre escrita científica neste trabalho, além da ajuda com a compra dos materiais.

Ao meu orientador, professor Eduardo Verri Liberado, pela oportunidade de desenvolver este trabalho e pela ajuda na escrita.

Aos amigos Caio Cortada, Christian Schuarcz, Giuliano Faria, Guilherme Gomes, João Paulo, Ricardo Bozollan e Thiago Colombari, pelo apoio durante minha formação e contribuição neste trabalho.

Ao Luigi Grandi, pela ajuda durante minha formação, esclarecimento de dúvidas sobre o PFC e auxílio na formatação.

À banca avaliadora e suplente: professores Leopoldo Lusquino, Luis Arenas e Eduardo Paciência.

Ao professor Everson Martins, pelo suporte nas questões relacionadas à coordenação do curso.

Ao meu primo Guilherme Apollinario, por ter me dado a estação de solda utilizada no projeto, permitindo seu desenvolvimento.

À Dinâmica Engenharia Júnior e seus membros, por proporcionar experiências como gerente de projetos e diretoria executiva.

Ao meu gestor Guilherme Kassis e colegas de trabalho, pelo apoio diário e pela oportunidade de aplicar meus conhecimentos.

Aos colegas de classe, pelos anos de convivência e aprendizado conjunto.

Aos professores, técnicos e funcionários do ICTS, pelo papel essencial em minha formação.

E por fim, à Universidade Estadual Paulista (UNESP), pela oportunidade de cursar Engenharia de Controle e Automação, essencial para meu crescimento pessoal e profissional.

## RESUMO

Com a necessidade de ação em relação à escassez dos recursos hídricos aliada aos conceitos de Internet das Coisas (IoT) e Energy Harvesting, este trabalho apresenta o desenvolvimento de um sistema de irrigação automática com monitoramento remoto. Esse sistema utiliza um microcontrolador ESP8266 D1 Mini para realizar a coleta de dados de umidade do solo e acionar automaticamente uma válvula solenóide presente no sistema de irrigação, liberando a água para o que está sendo cultivado. A comunicação entre a unidade de controle e o usuário ocorre pelo protocolo MQTT (Message Queuing Telemetry Transport), o qual permite que um aplicativo móvel desenvolvido em React Native receba esses dados e realize o controle do sistema de forma remota. Além disso, o projeto incorpora um painel solar e baterias recarregáveis de íon de lítio, garantindo sua operação mesmo em locais remotos sem acesso à rede elétrica. Os resultados demonstram que o sistema pode otimizar o uso da água, reduzindo desperdícios e proporcionando maior autonomia na gestão da irrigação. O uso da automação e do monitoramento remoto simplifica a manutenção e melhora a eficiência hídrica, tornando-o uma solução viável para aplicações agrícolas e domésticas.



Palavras-chave: Irrigação Automática; Monitoramento Remoto; IoT; ESP8266; Energy Harvesting; MQTT; React Native.

**ABSTRACT**

Considering the need for action regarding water resource scarcity, combined with the concepts of Internet of Things (IoT) and Energy Harvesting, this work presents the development of an automatic irrigation system with remote monitoring. This system utilizes an ESP8266 D1 Mini microcontroller to collect soil moisture data and automatically activate a solenoid valve in the irrigation system, releasing water to the cultivated area. The communication between the control unit and the user is carried out via the MQTT (Message Queuing Telemetry Transport) protocol, which allows a mobile application developed in React Native to receive this data and control the system remotely. Additionally, the project incorporates a solar panel and rechargeable lithium-ion batteries, ensuring its operation even in remote locations without access to the electrical grid. The results demonstrate that the system can optimize water usage, reduce waste, and provide greater autonomy in irrigation management. The use of automation and remote monitoring simplifies maintenance and improves water efficiency, making it a viable solution for both agricultural and domestic applications.

Keywords: Automatic Irrigation; Remote Monitoring; IoT; ESP8266; Energy Harvesting; MQTT; React Native.

## LISTA DE FIGURAS

## LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

| | | |
|---|---|---|
| API | - | Application Programming Interface |
| CB | - | Cell Broadcast |
| EEPROM | - | Electrically Erasable Programmable Read-Only Memory |
| GPIO | - | General Purpose Input/Output |
| GSM | - | Global System for Mobile Communications 2G |
| IoT | - | Internet of Things |
| iOS | - | iPhone Operating System |
| JSON | - | JavaScript Object Notation |
| LED | - | Light Emitting Diode |
| macOS | - | Macintosh Operating System |
| MQTT | - | Message Queuing Telemetry Transport |
| NTP | - | Network Time Protocol |
| PCB | - | Printed Circuit Board |
| Protobuf | - | Protocol Buffers |
| RTC | - | Real Time Clock |
| TCP | - | Transmission Control Protocol |
| TLS | - | Transport Layer Security |
| USB | - | Universal Serial Bus |
| Wi-Fi | - | Wireless Fidelity |

## LISTA DE SÍMBOLOS

| | | |
|---|---|---|
| i | corrente | *A* |
| L | indutância | *H* |
| t | tempo | *s* |
| V | tensão | *V* |

**SUMÁRIO**

# 1 INTRODUÇÃO

A escassez de recursos hídricos é um desafio global cada vez mais presente no dia a dia da população, com repercussões significativas em diversas áreas, especialmente nas atividades agrícolas. A demanda crescente por água, o desequilíbrio entre oferta e demanda e os efeitos das mudanças climáticas têm ampliado a pressão sobre os recursos hídricos, tornando essencial uma gestão mais eficiente e sustentável. (CETESB, 2023)

Na agricultura, em particular, onde a água é um recurso fundamental, a escassez pode resultar em perdas significativas de produção e impactar a segurança alimentar global. Nesse contexto, é fundamental desenvolver e implementar tecnologias e práticas que permitam o uso mais racional da água e contribuam para a mitigação dos efeitos da escassez hídrica.

Aproximadamente 70% da água do Brasil e do mundo é utilizada em atividades agrícolas, porém, cerca de 50% a 60% é desperdiçada devido à má gestão desse valioso recurso. Segundo Caio Bacci, Head de Marketing da Agrosmart, especializada em agricultura digital e manejo e monitoramento de irrigação, "Na maioria dos casos o agricultor não sabe quanto, quando e de que forma deve irrigar" (Redação Agrishow, 2019).

Diante dessa problemática, é evidente a necessidade de uma gestão sustentável e eficiente dos recursos hídricos, especialmente na agricultura. Utilizar estratégias inovadoras e tecnológicas se torna essencial para otimizar o uso da água, garantindo viabilidade econômica, social e ambiental das práticas agrícolas. Integrar sistemas de irrigação mais precisos, monitoramento em tempo real e técnicas de conservação de água são passos importantes nessa direção. Essas medidas não apenas reduzem o desperdício de água, mas também contribuem para a preservação dos ecossistemas e com a sustentabilidade das comunidades agrícolas.

Dessa maneira, é proposto o desenvolvimento de um sistema IoT (*Internet of Things*) de irrigação automatizada. Inicialmente, o sistema monitora a umidade do solo de uma cultura específica por meio de um sensor dedicado. Com base nas leituras do sensor, o sistema pode decidir automaticamente se deve abrir ou fechar a válvula solenóide responsável pelo fluxo de água no sistema de irrigação.

Adicionalmente, um aplicativo móvel é desenvolvido para possibilitar o monitoramento e o controle remoto da válvula. Através do aplicativo, os usuários podem acompanhar os níveis de umidade do solo em tempo real e ajustar as configurações de irrigação conforme necessário, tudo de forma remota e conveniente.

Por fim, pensando na utilização do sistema em locais remotos, o conceito de *Energy harvesting* é utilizado, conectando toda a alimentação do sistema a um sistema composto por uma placa solar e baterias recarregáveis, de modo a garantir o suprimento de energia inclusive no caso de ausência de luz. Um microcontrolador conectado à *internet* via Wi-Fi (*Wireless Fidelity*) é o responsável por fazer o monitoramento do sensor de umidade do solo, acionamento da válvula e envio dos dados para um *broker* MQTT (*Message Queuing Telemetry Transport*), de forma a serem consumidos pelo aplicativo de celular.

Assim, por meio desse protótipo, é esperado que se reduza o desperdício de recursos hídricos e mão de obra, devido a automatização, além de reduzir os impactos ambientais da agricultura, possibilitando a instalação independente da rede pelo recurso de *Energy Harvesting*.

Dentro desse contexto, o Capítulo 2 apresenta uma revisão bibliográfica sobre outras soluções para a problemática em questão. Em seguida, o Capítulo 3 aborda a fundamentação teórica, enquanto o Capítulo 4 descreve os materiais e métodos utilizados. O Capítulo 5 traz os resultados e as discussões, e, finalmente, o Capítulo 6 apresenta as conclusões.

# 2 REVISÃO BIBLIOGRÁFICA

Quando o assunto é irrigação, abrange desde pessoas que possuem um jardim em casa, até grandes campos agrícolas. Um sistema IoT de irrigação economiza tempo e recursos, sendo uma solução eficiente para ambas as necessidades. Ele simplifica o processo de rega, garantindo que as plantas recebam a quantidade certa de água, quando necessário. Essa tecnologia proporciona uma gestão mais inteligente e econômica da água, tornando a irrigação mais fácil e eficaz.

Atualmente na agricultura brasileira, há diversos fatores que implicam no desperdício de água. Um deles é a aplicação de água em excesso realizada em horários equivocados, já que em determinados momentos do dia a velocidade do vento é alta, somando com a temperatura elevada e umidade do ar baixa, causa ineficiência na irrigação (PORTAL DO GOVERNO, 2006).

Além disso, quando não há um sistema controlado, pode ocorrer de irrigar mais do que a planta necessita, o que ocorre facilmente quando controlado manualmente. Com isso ocorre a lixiviação de nutrientes do solo, o aparecimento de pragas e doenças de forma mais frequente, resultando até na morte da planta (Maneje Bem, 2018). Segundo a Agência Nacional de Águas e Saneamento Básico (ANA), essa falta de precisão na irrigação faz com que se exceda 40% da água necessária para as culturas, resultando em prejuízos significativos para a produção (EQUIPE RAKS, 2021).

Em 2016, o Ministério da Agricultura, Pecuária e Abastecimento (Mapa), previu uma expansão da área irrigada de 6,2 milhões para 11,2 milhões até 2026, demonstrando um crescente mercado na área (SILVA, 2018).

Algumas pesquisas indicaram que sistemas de irrigação automatizados baseados em IoT podem otimizar o uso de água e energia na agricultura, principalmente em regiões remotas sem acesso à rede elétrica. Pernapati (2018) sugere um sistema de irrigação baseado em IoT, utilizando sensores sem fio para monitoramento das condições ambientais e do solo. Esse sistema emprega o protocolo MQTT para realizar uma comunicação eficiente, permitindo a ativação automática da irrigação quando a umidade do solo atinge níveis baixos. Além disso, há sensores ultrassônicos que verificam o nível de água no reservatório, evitando desperdícios. Os resultados demonstraram que a abordagem proposta reduz significativamente o consumo de água e minimiza a intervenção humana, tornando-se uma solução viável para a modernização da agricultura.

Já Islam *et al.* (2019) propuseram um sistema de irrigação IoT utilizando sensores de umidade do solo, energia solar e o protocolo MQTT para comunicação em tempo real. Esse sistema monitora automaticamente a umidade do solo e aciona a bomba de irrigação conforme necessário, enviando mensagens via GSM (*Global System for Mobile Communications 2G*) aos agricultores. Os resultados mostraram que a combinação de IoT, MQTT e energia solar reduz custos operacionais, melhorando a eficiência hídrica e minimizando a necessidade de intervenção manual, tornando-se uma solução viável e sustentável para áreas agrícolas isoladas.

No entanto, a proposta desse trabalho é uma solução que integre o sistema de medição de umidade do solo, acionamento de válvulas e *Energy Harvesting* em um único aplicativo. A integração desses componentes ofereceria uma abordagem mais abrangente e eficiente para a gestão da irrigação, proporcionando maior controle e economia de recursos.

Essa integração permitiria que os usuários monitorem e controlem todos os aspectos da irrigação de forma centralizada e conveniente, por meio de um único aplicativo. Além disso, a utilização de *Energy Harvesting* garantiria uma fonte de energia sustentável e econômica para alimentar o sistema, tornando-o mais autônomo e eficiente.

Portanto, há uma clara oportunidade no mercado para o desenvolvimento de soluções que integrem esses diferentes elementos em um único sistema, oferecendo uma abordagem mais completa e eficaz para a gestão da irrigação na agricultura brasileira.

# 3 FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento desse projeto, foi necessário o conhecimento em diversas áreas que abrangem o curso de Engenharia de Controle e Automação, desde conceitos de *Energy Harvesting*, *Internet* das coisas até comunicação de dispositivos utilizando comunicação MQTT.

## 3.1 Energy Harvesting

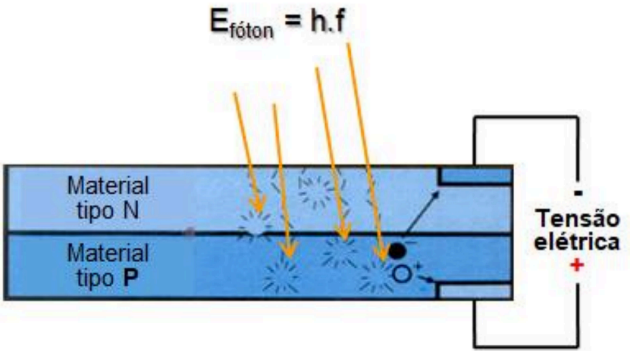
O conceito de *Energy harvesting* consiste em capturar pequenas quantidades de energia disponível no ambiente e sua conversão em energia elétrica utilizável, geralmente para alimentar dispositivos de baixa potência. Essa abordagem é amplamente utilizada em aplicações que consistem em monitoramento remoto e sistemas autônomos, pois permite prolongar a vida útil das baterias ou até mesmo eliminá-las da aplicação, permitindo que o sistema tenha uma operação contínua e mais sustentável (HARB, 2011; SANTANA *et al.*, 2024).

Há diversas fontes de energia que podem ser exploradas em sistemas de *Energy Harvesting*, incluindo vibrações mecânicas, gradientes térmicos, radiação eletromagnética e energia solar. No entanto, a energia solar destaca-se quando se trata de projetos de irrigação automática, devido à sua disponibilidade constante em áreas agrícolas (HARB, 2011).

A conversão da energia solar em energia elétrica por meio de painéis fotovoltaicos, é uma das principais técnicas utilizadas atualmente de *harvesting*, se tornando também uma das formas mais eficientes. Esse tipo de energia natural oferece uma maior densidade energética comparada com outras fontes de energia, como piezoelétricos ou termelétricos (BHUVANESWARI *et al.*, 2009).

Seu princípio de funcionamento consiste no efeito fotovoltaico, efeito esse no qual é gerada uma diferença de potencial entre os terminais de um dispositivo semicondutor quando o mesmo é exposto aos fótons presentes na luz (MICHA, 2017). Pode-se observar a representação desse efeito na Figura 1.

Figura 1 - Representação do efeito fotovoltaico




Fonte: Daniel Micha (2017).

Com a incidência desses fótons presentes na luz solar sobre os semicondutores, essas partículas com energia suficiente são absorvidos pelo material semicondutor. Essa energia excedente faz com que os elétrons da camada de valência do material semicondutor sejam excitados para a banda de condução, deixando uma "lacuna" na camada de valência (MICHA, 2017).

Considerando uma junção P-N como na imagem, quando há a lacuna, o campo elétrico interno criado pela junção atua separando as cargas. O elétron excitado é direcionado para a região tipo N, enquanto a lacuna (ou buraco) migra para a região tipo P. Quando conectado um circuito externo entre as regiões do tipo P e N, os elétrons fluem da região N para a região P através desse circuito, gerando uma corrente elétrica contínua (MICHA, 2017).

Considerando sistemas *off-grid* (independentes da rede elétrica), o uso de painéis solares permite uma recarga contínua das baterias durante o dia, reduzindo a frequente necessidade de manutenção, promovendo a autonomia energética. A energia capturada é armazenada em baterias recarregáveis, como baterias de íon de lítio, que normalmente são utilizadas devido sua alta densidade de energia e vida útil prolongada (BHUVANESWARI *et al*., 2009).

Nesse contexto, a adoção de *Energy Harvesting* na irrigação automática proporciona um sistema mais eficiente, sustentável e autônomo, eliminando a dependência da rede elétrica e reduzindo a necessidade de intervenção humana. O desenvolvimento de circuitos de gerenciamento de energia e estratégias eficientes de armazenamento energético são fundamentais para maximizar o desempenho dessa tecnologia.

## 3.2 *Internet* das Coisas

O conceito de *Internet* das Coisas (IoT), é amplamente utilizado na quarta revolução industrial. Ele consiste na interconexão de dispositivos físicos por meio da *internet*, possibilitando a coleta, transmissão e análise de dados em tempo real. Esse conceito foi proposto por Kevin Ashton em 1999, sendo aplicado desde então em diversas áreas, como por exemplo, processos de automação agrícola de irrigação inteligente (GOKHALE *et al*., 2018).

Dessa forma, é possível realizar a comunicação entre sensores, atuadores e sistemas computacionais de forma autônoma e eficiente, formando um ecossistema entre esses dispositivos. Esse ecossistema possui uma arquitetura típica de sistemas IoT, possuindo sensores para coleta de dados, microcontrolador para processamento de informações, um módulo de comunicação para transmissão de dados, e por fim, uma interface para visualização desses dados e realização do controle de forma remota (SRIVASTAVA *et al*., 2018).

Nesse contexto, é válido citar como um dos principais microcontroladores utilizados em projetos IoT sendo o ESP8266. Esse microcontrolador possui conectividade Wi-Fi integrada, além de seu custo ser baixo, facilita sua ampla utilização nesse tipo de projeto. Principalmente por conta da sua conectividade sem fio já embutida, esse dispositivo permite a comunicação de sensores e atuadores com uma interface para o usuário por meio de protocolos de comunicação, como o MQTT (SRIVASTAVA *et al*., 2018).

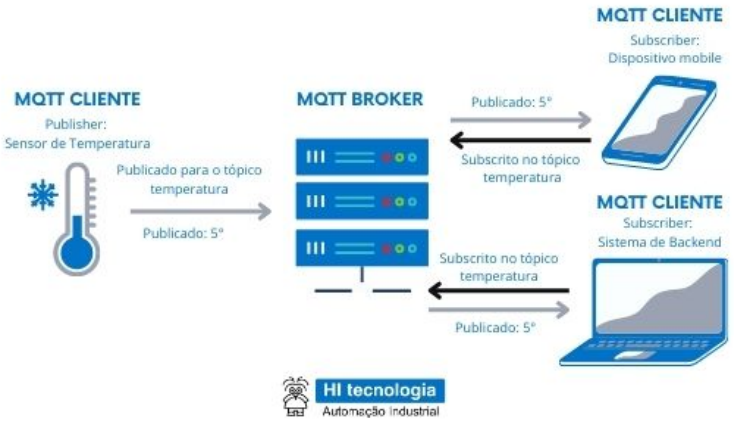
### 3.2.1 Protocolo MQTT

O MQTT é um protocolo projetado para redes com baixa largura de banda e consumo reduzido de energia, sendo ideal para sistemas IoT autônomos (YASSEIN *et al*., 2017). Esse protocolo de comunicação implementa o modelo de publicação/assinatura, que é composto de clientes e agentes. O cliente MQTT é um dispositivo que pode tanto atuar como publicador, enviar mensagens, ou como destinatário se receber. Ou seja, qualquer dispositivo que se comunique por MQTT em uma rede pode ser chamado de cliente MQTT.

Já o agente, também conhecido como *broker*, é o sistema *backend* que gerencia as mensagens entre diferentes clientes. Ele também é responsável por gerenciar a conexão e garantir que a mensagem publicada por um cliente consiga chegar para os assinantes interessados na mensagem.

As mensagens são organizadas em tópicos hierárquicos, permitindo que os clientes publiquem informações em um tópico específico, e todos os dispositivos assinantes desse tópico recebam os dados automaticamente. Esse modelo é ilustrado na Figura 2.

Figura 2 - Exemplo de funcionamento protocolo MQTT




Fonte: Hi Tecnologia (2021).

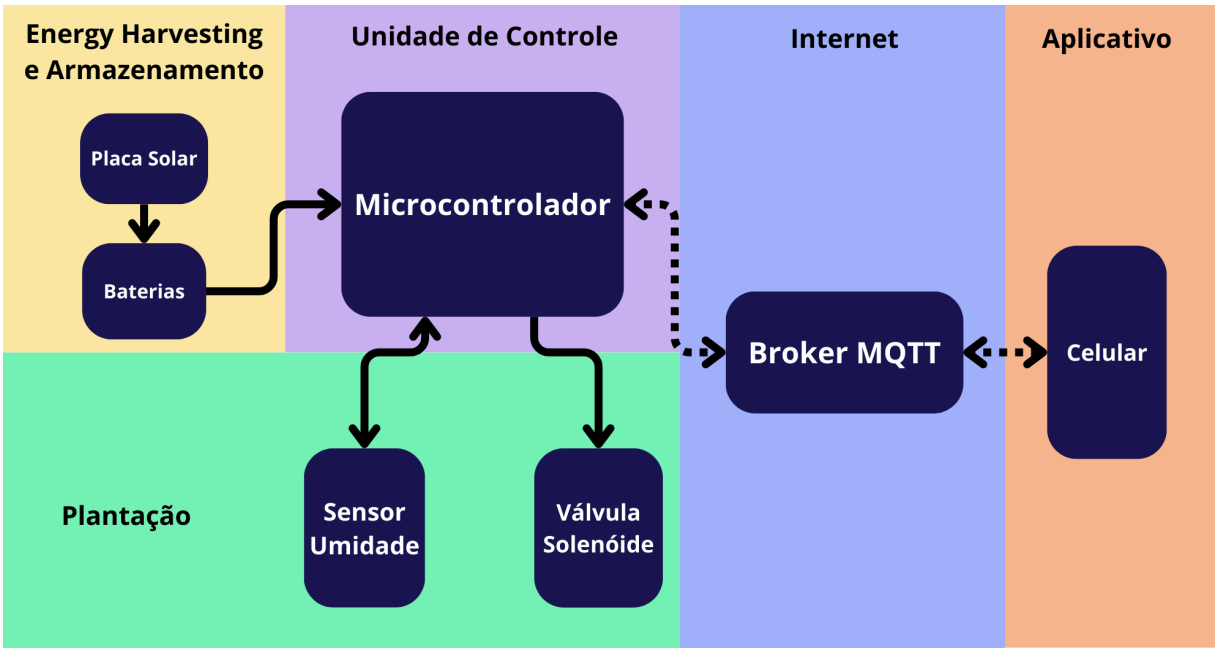
Dessa forma, a utilização do MQTT em conjunto com o ESP8266 proporciona uma aplicação IoT robusta, escalável e de fácil implementação. Ele é capaz de realizar a troca de informações entre os dispositivos físicos e a interface de controle em tempo real e com um baixo custo energético, permitindo que os usuários controlem e acompanhem o sistema por dispositivos móveis ou plataformas *web*, aumentando a eficiência e praticidade do processo. (YASSEIN et. al, 2017; SRIVASTAVA et. al, 2018).

# 4 MATERIAIS E MÉTODOS

Para o desenvolvimento do projeto, foi adotada a construção de um sistema de *Energy Harvesting* utilizando luz solar como fonte primária de energia elétrica, à qual é regulada por meio de um conversor de corrente contínua (CC-CC). Além disso, foi especificada uma unidade de controle baseada em um microcontrolador ESP8266 D1 Mini, responsável pela aquisição de dados do sensor de umidade do solo e pelo acionamento da válvula solenóide.

O projeto também envolveu o desenvolvimento do *software* embarcado, incluindo a lógica para envio e recebimento de informações via *broker* MQTT, além da criação do aplicativo móvel IrrigaHub, que permite o controle remoto do sistema. O fluxograma geral do projeto pode ser visualizado na Figura 3.

Figura 3 – Fluxograma idealizado do projeto




Fonte: Autoria própria (2025).

No fluxograma, a direção das setas indicam o fluxo de energia ou informações, sendo que as linhas inteiras representam conexões físicas e as pontilhadas indicam comunicação via rede.

Na construção do sistema de captação de energia, foi utilizada uma placa solar em conjunto com baterias de lítio recarregáveis. Durante o dia a placa é capaz de fornecer energia para que seja realizado o carregamento dessas baterias, para que sejam utilizadas pela carga em momentos de pouca ou nenhuma luz.

A alimentação do sistema ocorre por um conversor CC-CC Boost, cuja finalidade é elevar o nível de tensão proveniente da bateria, de 3,7 V para 5 V, sem comprometer demasiadamente a potência fornecida, devido à eficiência desses conversores. Dessa forma, o sistema é alimentado com uma tensão adequada para seu funcionamento.

A unidade de controle, baseada no ESP8266 D1 Mini, é responsável por monitorar a umidade do solo e acionar a válvula solenóide por meio de um circuito de chave eletrônica. Além disso, conta com botões físicos para reinicialização e redefinição da rede Wi-Fi.

O microcontrolador cria uma rede Wi-Fi temporária para configuração inicial, onde o usuário pode conectar-se via dispositivo móvel e definir a rede que o dispositivo se conectará. Após configurado, o ESP8266 se conecta a um *broker* MQTT público, utilizando um nome definido pelo usuário como identificador para publicação de mensagens.

O aplicativo IrrigaHub permite o monitoramento remoto da umidade do solo e o controle da irrigação, seja por ativação manual, tempo programado ou nível de umidade desejado. A comunicação entre o dispositivo físico e o aplicativo ocorre via MQTT, garantindo sincronização eficiente e em tempo real.

## 4.1 Componentes Físicos

Foi realizado um dimensionamento tomando como base o consumo médio do ESP, capacidade de armazenamento de bateria e potência da placa solar. Além disso, conversores de energia foram empregados para atender as necessidades de tensões distintas do circuito.

Conectados ao ESP8266 estão a válvula solenóide, permitindo o controle do fluxo da água, e o sensor de umidade do solo que mede a condutividade do solo que ocorre pela umidade presente.

Nota-se que os componentes foram adquiridos com um orçamento limitado, dessa forma, há componentes mais indicados para essa aplicação disponíveis no mercado, mas com o objetivo de reduzir custos do projeto, foram utilizadas opções alternativas.

### 4.1.1 Placa Solar

É utilizada uma placa solar de 3 W de potência, com 12 V de tensão, para suprir o consumo do sistema. O sistema possui um consumo basal de corrente de 80 mA a 5 V, resultando em um consumo de aproximadamente 0,4 W. No entanto, quando a válvula solenóide é ativada, a corrente sobe para 300 mA, elevando o consumo para 1,5 W.

Assim, com uma potência nominal de 3 W, a placa solar é capaz de atender ao consumo máximo do sistema e ainda fornecer energia excedente, quando o sistema de irrigação não é acionado, para recarregar a bateria quando exposta a luz solar suficiente para gerar sua potência nominal, garantindo autonomia durante períodos de baixa luminosidade ou ausência de luz solar.

### 4.1.2 Bateria

O sistema utiliza um conjunto de duas baterias recarregáveis de íon de lítio de 3,7 V conectadas em paralelo, permitindo um maior fornecimento de corrente sem alterar a tensão. Cada bateria possui uma capacidade de 2600 mAh, totalizando 5200 mAh para as duas baterias. Com a carga máxima, o sistema pode operar por aproximadamente 65 horas em estado basal ou 17 horas com a válvula continuamente ativada, sem necessidade de recarga.

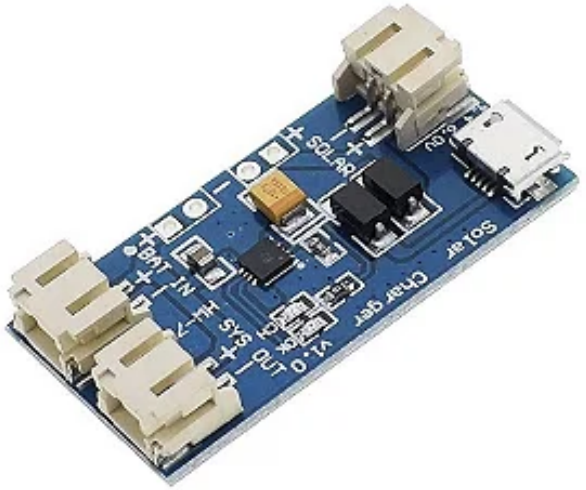
O dimensionamento levou em consideração que o sistema necessita operar por no mínimo 24 horas em consumo máximo de energia sem recarregamento, garantindo que o protótipo resista ao menos dois ciclos de 12 horas sem contato com luz solar, considerando a possibilidade de um dia nublado e um período noturno, sendo que em ambos não haveria o carregamento das baterias.

### 4.1.3 Módulo Carregador Solar para Bateria de Lítio

Para o gerenciamento da bateria, é utilizado o módulo carregador solar CN3065, que consegue gerenciar a carga de baterias de lítio via painel solar, gerenciando quando a bateria precisa de carga e quando enviá-la ao sistema. Sua corrente máxima de carregamento é 500 mA, com uma entrada com tensão mínima de 4,4 V e máxima de 6 V para a célula solar.

A tensão é reduzida para a tensão da bateria, possuindo então uma saída para carga de 3,7 V, sendo essa a mesma da bateria, visto que essas duas saídas estão conectadas em paralelo. Tal dispositivo pode ser visualizado na Figura 4.

Figura 4 - Módulo carregador solar para baterias de lítio CN3065



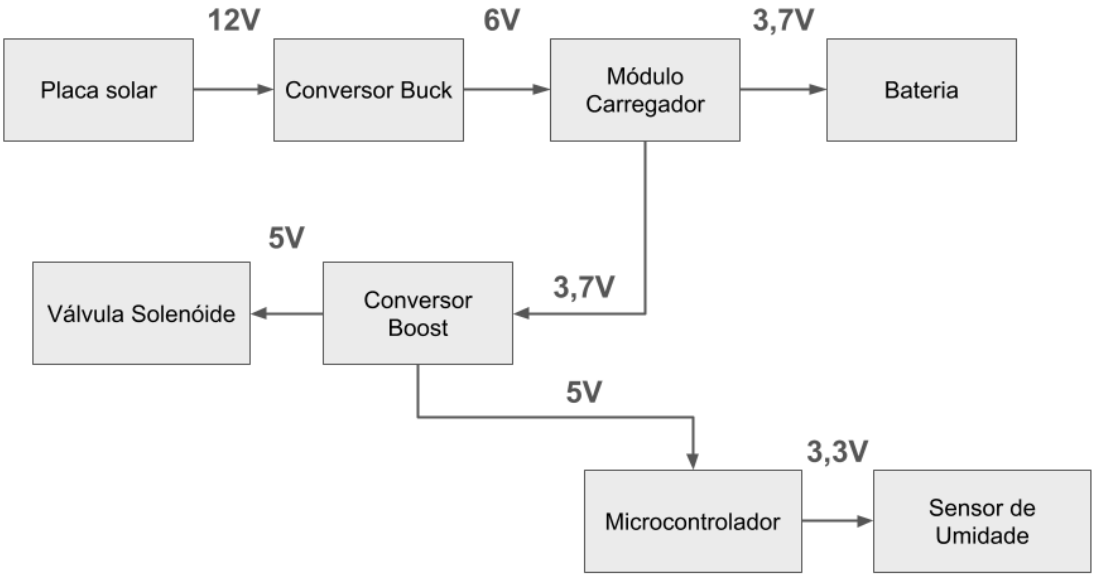

Fonte: Eletrogate (2025).

### 4.1.4 Conversores CC-CC

Devido às variadas tensões requeridas pelo sistema, são utilizados conversores de corrente contínua (CC-CC) para ajustar os níveis de tensão, prevenindo sobretensão ou subtensão e garantindo o funcionamento adequado dos componentes. Para isso, foram empregados conversores Buck e Boost, conhecidos por sua alta eficiência (RASHID, 2014).

O conversor Buck reduz a tensão da saída do painel solar (cujo valor máximo é 12 V) para a faixa adequada de entrada do gerenciador de carga da bateria (4,4 V - 6 V). Já o conversor Boost eleva a tensão da bateria de 3,7 V para 5 V, permitindo a alimentação do ESP8266 e da válvula solenóide. Dessa forma, a Figura 5 ilustra todas as conversões de nível de tensão apresentadas no projeto.

Figura 5 - Diagrama de conversão do nível de tensão

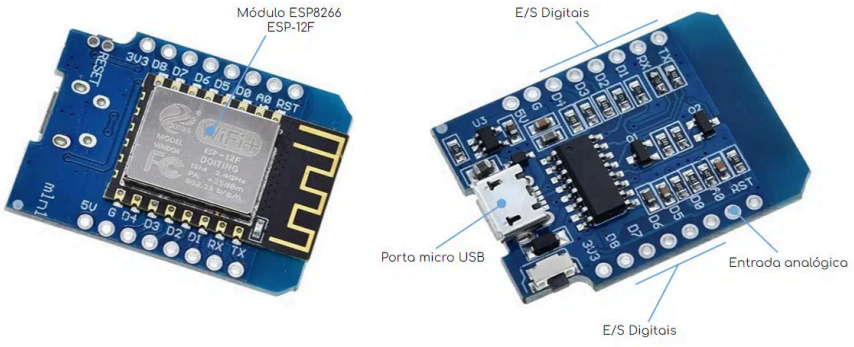



Fonte: Autoria própria (2025).

### 4.1.5 Unidade de Controle

A unidade de controle é composta por um módulo ESP8266 D1 Mini, que possui onze pinos digitais de entrada e saída, e um pino de entrada analógica, que é utilizado para a leitura do sensor de umidade. O módulo também conta com uma porta micro USB (*Universal Serial Bus*) que permite a alimentação e comunicação com o computador, como observado na Figura 6.

Figura 6 - Entradas e controlador da placa ESP8266 D1 Mini



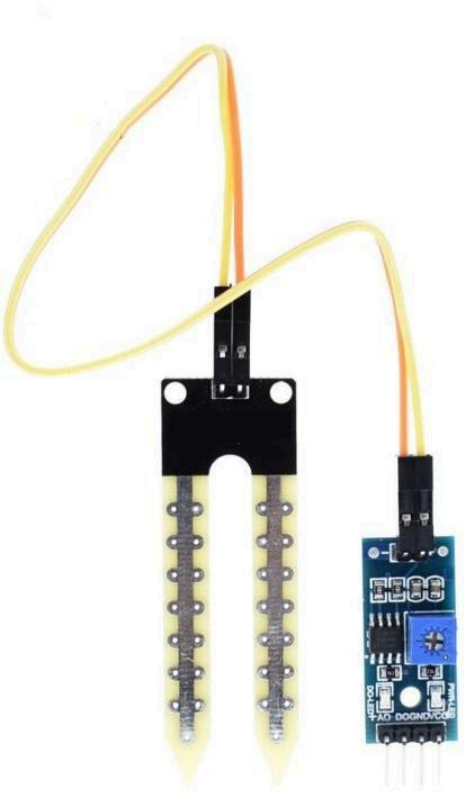

Fonte: MakerHero (2022).

Suas portas operam em nível lógico de 3,3 V, porém, o módulo possui um pino de alimentação que suporta 5 V, por conta de um regulador de tensão interno, e outro que distribui alimentação em 3,3 V, possibilitando os botões e sensor operar com o mesmo nível lógico do controlador, enquanto recebe alimentação de 5 V, garantindo assim a compatibilidade e estabilidade do circuito.

### 4.1.6 Sensor de Umidade

O sensor de umidade do solo, também conhecido como higrômetro, é um dispositivo composto do módulo HW-080 e o comparador HW-103. Esse sensor pode ser visualizado na Figura 7.

Figura 7 - Módulo HW-080 à esquerda e comparador HW-103 à direita



Fonte: Curto Circuito (2025).

Esse dispositivo é um sensor resistivo, ou seja, ele mensura a resistência elétrica entre as duas hastes. Quando colocado no solo, uma baixa resistência indica alta umidade do solo, enquanto uma alta resistência, que está seco.

Essa informação é então passada ao comparador, que por um divisor de tensão, compara a tensão resultante com uma de referência e a transforma em um sinal analógico, cuja intensidade depende da umidade do solo: quanto maior a leitura, mais seco está o solo. Essa informação é então processada pelo microcontrolador, permitindo a tomada de decisões no sistema de irrigação.

### 4.1.7 Válvula Solenoide

A válvula solenóide é um dispositivo eletromecânico utilizado para controlar o fluxo de fluidos, permitindo sua abertura ou fechamento por meio de um comando elétrico.

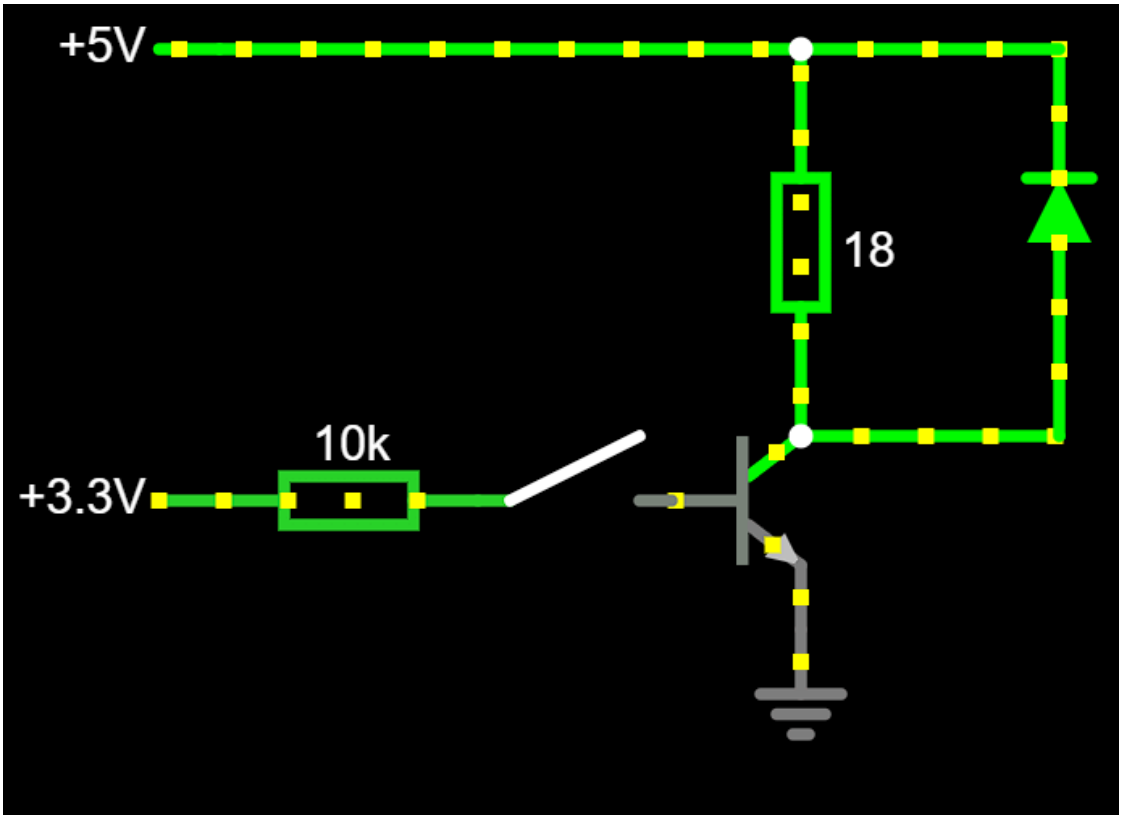
Seu princípio de funcionamento é utilizar ímãs permanentes em conjunto com um eletroímã. A válvula utilizada no projeto é normalmente fechada, ou seja, impede a passagem do fluido em seu estado inativo. Isso ocorre devido à repulsão magnética entre dois ímãs de mesma polaridade, posicionados no interior da válvula.

Quando uma corrente elétrica percorre o enrolamento do eletroímã, é gerada uma força eletromagnética de acordo com o fluxo da corrente. Dessa forma, a força eletromagnética é capaz de superar a repulsão e atrair o imã, permitindo a passagem do fluido pelos canais da válvula solenóide.

Porém, como a válvula consome uma quantidade considerável de corrente elétrica, as portas do ESP não conseguem prover essa corrente diretamente. Para solucionar esse problema, utilizou-se um transistor NPN BC548 como chave eletrônica. A válvula empregada opera em 3,7 V, sendo alimentada pelo mesmo nó de tensão da saída do conversor Boost. Há uma queda de aproximadamente 1V da tensão no sistema de ativação, garantindo que a tensão que chega na válvula seja segura para operação.

No circuito, o emissor do transistor é aterrado, enquanto o coletor está conectado à válvula solenóide. O microcontrolador envia um sinal lógico à base do transistor, saturando-o e permitindo que a corrente flua, acionando a válvula. Pode-se observar a representação do circuito na Figura 8.

Figura 8 - Esquema do sistema de ativação da válvula




Fonte: Autoria própria (2025).

Nota-se que nessa representação, o conjunto do resistor de 10 kΩ e chave representa o sinal emitido pela porta digital do ESP, enquanto o resistor de 18 Ω representa a impedância dos enrolamentos da válvula solenóide.

Além disso, foi incorporado um diodo de roda-livre em paralelo reverso com a válvula para evitar picos de tensão que poderiam danificar o transistor. Estes picos ocorrem pois a válvula solenóide possui um comportamento indutivo. Segundo a equação (1), quanto maior a variação da corrente, representada por sua derivada em função do tempo, maior será a tensão induzida no indutor ($V$).

$$V = L \cdot \frac{di}{dt} \quad (1)$$

Dessa forma, ao interromper abruptamente a corrente no transistor com o desligamento da válvula, o diodo fornece um caminho para a circulação residual da corrente, dissipando a energia de forma segura.

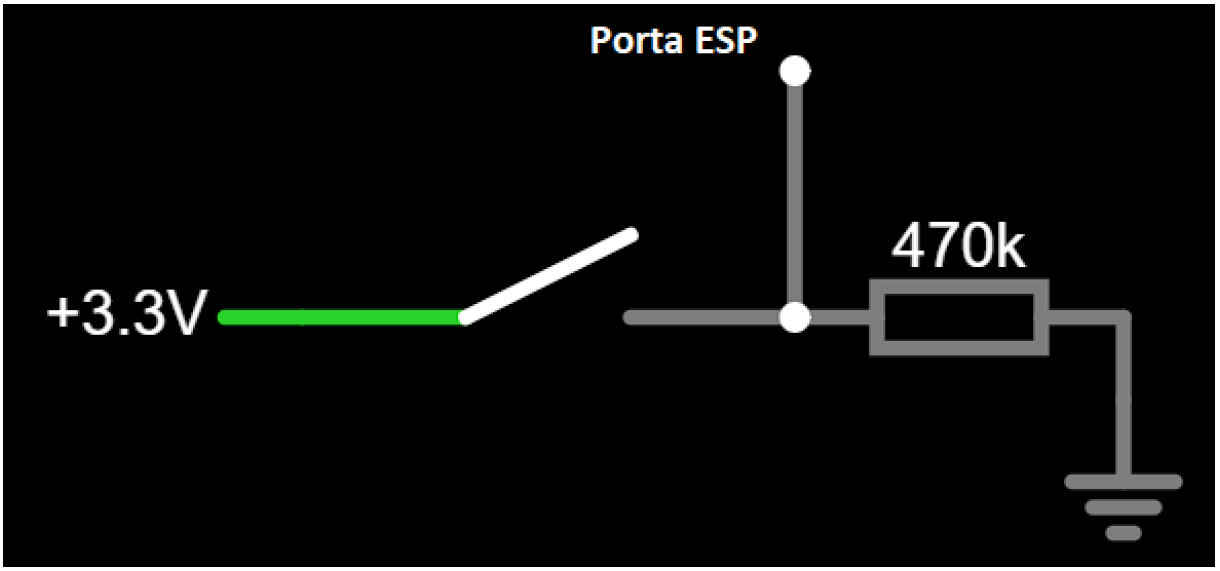
No projeto, devido à falta de um diodo convencional, foi empregado um LED (*Light Emitting Diode*), que também atua como um diodo emissor de luz, cumprindo a mesma função de proteção.

### 4.1.8 Botões

Foram utilizados dois botões no sistema para as funções de reinicialização e redefinição da rede do microcontrolador. Para isso, ambos foram colocados na disposição *pull-down*, ou seja, garantindo que a entrada digital do ESP8266 permaneça em nível lógico baixo até que o botão seja pressionado.

Nessa configuração, um dos terminais do botão está conectado a 3.3 V (nível lógico alto), enquanto o outro terminal está ligado à entrada digital do ESP, que, por sua vez, é mantida em nível baixo por meio de um resistor de 470 kΩ. Essa configuração pode ser observada na Figura 9.

Figura 9 - Esquema do sistema de ativação da válvula




Fonte: Autoria própria (2025).

Dessa forma, quando o botão for pressionado, o circuito é fechado, e a porta digital do ESP8266 recebe um nível lógico alto. Quando solto, o resistor garante que o sinal retorne ao nível baixo, evitando leituras instáveis no microcontrolador.

O resistor de 470 kΩ tem como função limitar a corrente, visto que para a leitura do microcontrolador basta detectar a mudança de estado lógico para processar a entrada do sinal, dispensando uma corrente elevada.

## 4.2 Arquitetura *Broke*r MQTT

Para possibilitar a conexão entre o dispositivo e o aplicativo móvel, foi escolhido o protocolo de comunicação MQTT. Essa escolha se deve ao seu baixo consumo de largura de banda e energia, além de ser ideal para comunicação em tempo real em projetos IoT. Outra vantagem do protocolo é a disponibilidade de *brokers* gratuitos na *internet*, eliminando a necessidade de hospedagem de um servidor próprio.

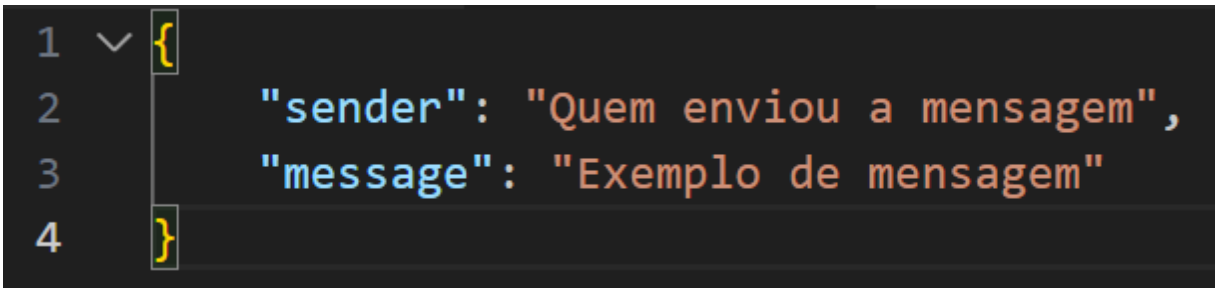
Dessa forma, foi escolhido o *broker* gratuito HiveMQ. Como o projeto ainda está na fase de protótipo, não há necessidade de investir em um *broker* pago, que garante o tempo em atividade e baixa latência constante. Sendo um *broker* público, foi escolhido um tópico base para filtrar as mensagens, evitando que outros usuários utilizem os mesmos tópicos que o sistema. Em relação à segurança, não foi utilizado a criptografia TLS (*Transport Layer Security*) nesse primeiro momento, porém, seria possível aplicá-la mesmo sendo um *broker* público.

### 4.2.1 Estrutura da Mensagem

Para padronizar a comunicação, optou-se por trabalhar com mensagens estruturadas no formato JSON (*JavaScript Object Notation*), amplamente utilizado para a comunicação na *internet*. Nota-se que esse formato possui um *overhead*, quantidade adicional de recursos utilizados durante a transmissão de dados, maior que formatos binários compactos, como CB (*Cell Broadcast*) ou *Protobuf (Protocol Buffers)*. Sendo assim recomendado que em caso da utilização de vários usuários simultâneos, seja implementado algum formato com *overhead* menor, já que o JSON pode impactar sistemas com restrições de energia ou largura de banda.

Dessa forma, todas as mensagens trocadas entre o aplicativo e o microcontrolador seguem essa estrutura, como demonstrada na Figura 10.

Figura 10 - Exemplo do formato da mensagem enviada ao *broker*



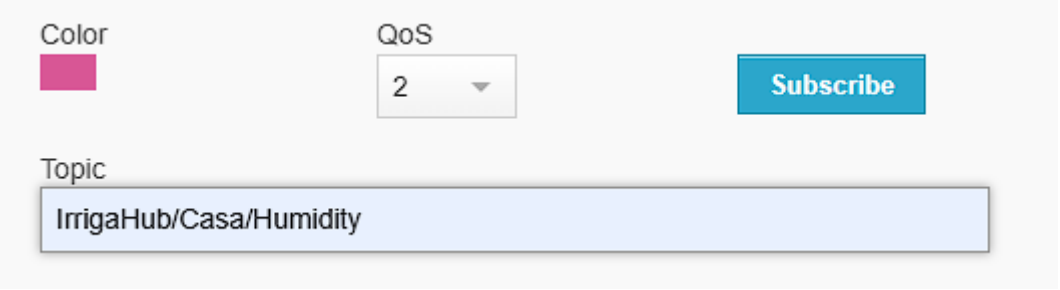

Fonte: Autoria própria (2025).

O campo *sender* possui dois valores possíveis, sendo eles “esp” ou “app”, permitindo então que o remetente seja identificado pelo destinatário. Essa estrutura possibilita que, dentro de um mesmo tópico, as mensagens sejam filtradas corretamente, garantindo que o dispositivo físico não execute ações destinadas ao aplicativo e vice-versa.

### 4.2.2 Estrutura dos Tópicos

Como o *broker* é público, foi definida uma estrutura de três níveis para garantir a sincronização entre o aplicativo e o dispositivo físico. O primeiro nível representa o nome do aplicativo, já o segundo identifica o dispositivo físico, e o terceiro é referente à funcionalidade do tópico. Essa estrutura é ilustrada na Figura 11.

Figura 11 - Exemplo de tópico para verificação da umidade do dispositivo “Casa”



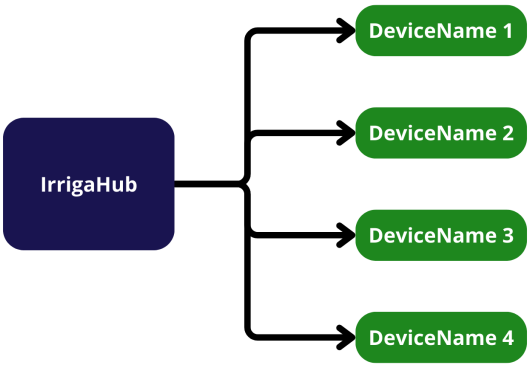

Fonte: Autoria própria (2025).

Para o primeiro nível, foi definido o nome“IrrigaHub”, nome do aplicativo. Esse nome é imutável, independentemente do usuário ou do dispositivo utilizado.

Já o segundo nível é variável e corresponde ao nome dado ao dispositivo, que deve ser configurado tanto na unidade de controle quanto no aplicativo móvel. Dessa forma, caso

ambos estejam com o mesmo nome, a comunicação ocorre corretamente, pois o dispositivo e o aplicativo estarão inscritos e publicarão mensagens nos mesmos tópicos, como observado na Figura 12.

Figura 12 - Representação da conectividade de vários dispositivos no *broker*

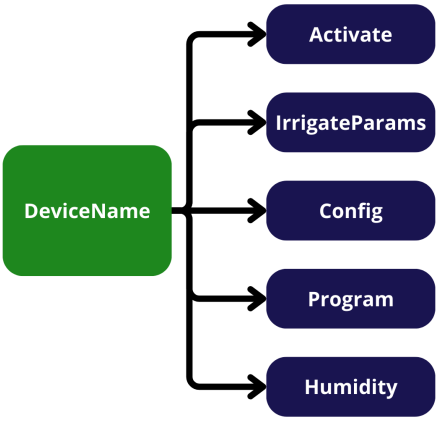



Fonte: Autoria própria (2025).

Essa estrutura possibilita a escalabilidade do sistema, permitindo a conexão de múltiplos dispositivos a um único dispositivo móvel. Como cada dispositivo possui um nome único, o sistema pode gerenciar vários tópicos simultaneamente sem conflitos.

Após esse nível, há um terceiro nível com tópicos relevantes para a funcionalidade do sistema, garantindo que cada dispositivo envie e receba apenas as informações necessárias. A Figura 13 apresenta um diagrama com os principais subtópicos utilizados.

Figura 13 - Representação dos principais subtópicos




Fonte: Autoria própria (2025).

Os principais subtópicos utilizados são:

- *Activate*: Recebe mensagens de ativação e inicia o sistema de irrigação com as configurações previamente definidas no controlador.
- *IrrigateParams*: Recebe parâmetros de configuração da irrigação, como duração ou umidade mínima desejada para interromper a irrigação.
- *Config*: Recebe comandos de configuração do sistema, como reinicialização do dispositivo ou redefinição das credenciais de Wi-Fi. Essa funcionalidade permite ajustes remotos sem a necessidade de acesso físico ao *hardware*.
- *Program*: Responsável pela programação da irrigação, permitindo o envio de parâmetros como dias da semana, horários e níveis de umidade para ativação da válvula solenóide.
- *Humidity*: Canal exclusivo para o envio de leituras do sensor de umidade do solo em tempo real.

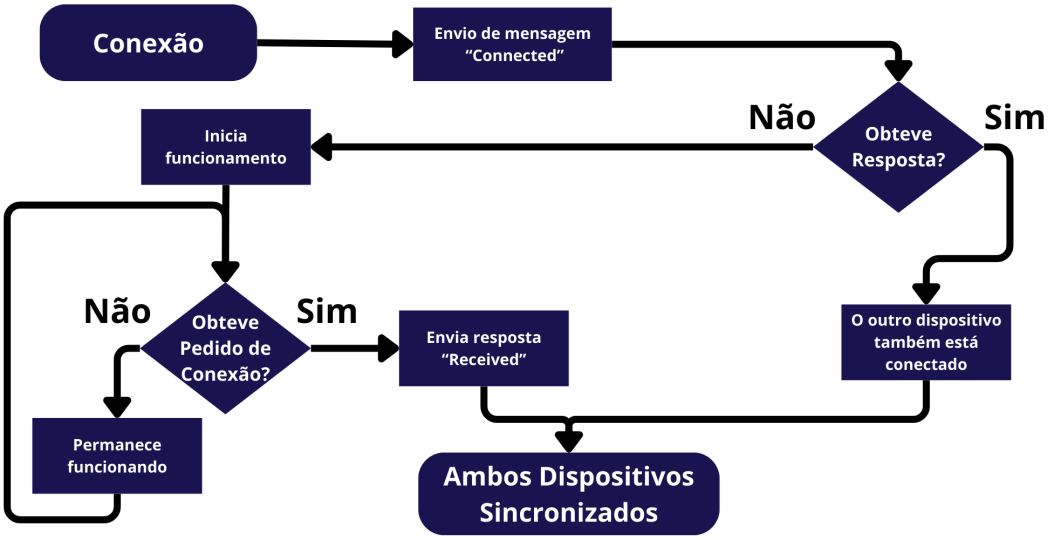
Essa estrutura modular garante a organização e eficiência na comunicação entre os dispositivos, tornando o sistema flexível e escalável para diferentes aplicações.

### 4.2.3 Lógica de Reconhecimento de Conexão

Para garantir o reconhecimento da conexão entre os dispositivos, seja o microcontrolador físico ou o aplicativo móvel, foi desenvolvida uma lógica específica. Embora ambos os dispositivos devam estar conectados simultaneamente para determinadas funcionalidades, eles podem operar de forma independente. Por exemplo, o dispositivo pode executar a irrigação mesmo que o aplicativo esteja desconectado, caso uma condição pré-programada seja atingida. Dessa forma, a conexão contínua do aplicativo não é um requisito absoluto.

Porém, diversas funcionalidades só ocorrem quando ambos os dispositivos estão conectados. Por isso, foi implementado um mecanismo de reconhecimento de conexão baseado no tópico central do dispositivo no MQTT, operando no segundo nível da hierarquia. O diagrama de blocos representando essa lógica está ilustrado na Figura 14.

Figura 14 - Diagrama de blocos da lógica utilizada para reconhecimento de conexão




Fonte: Autoria própria (2025).

A lógica funciona da seguinte forma: sempre que um dispositivo se conecta ao *broker* MQTT, ele envia uma mensagem de confirmação para o tópico central, por exemplo, *IrrigaHub/Casa*, indicando sua conexão. Caso ele receba uma resposta nesse mesmo tópico, ambos os dispositivos reconhecem que estão conectados simultaneamente.

Se essa resposta não for recebida, o dispositivo ou o aplicativo continua operando normalmente, sempre monitorando possíveis mensagens do outro dispositivo. Caso um dispositivo detecte um pedido de conexão, ele responde automaticamente, confirmando a conexão entre ambos.

Esse sistema garante que os dispositivos identifiquem quando estão conectados simultaneamente, permitindo o funcionamento de recursos que dependem dessa comunicação, como o envio da programação de irrigação do aplicativo para a unidade de controle na plantação.

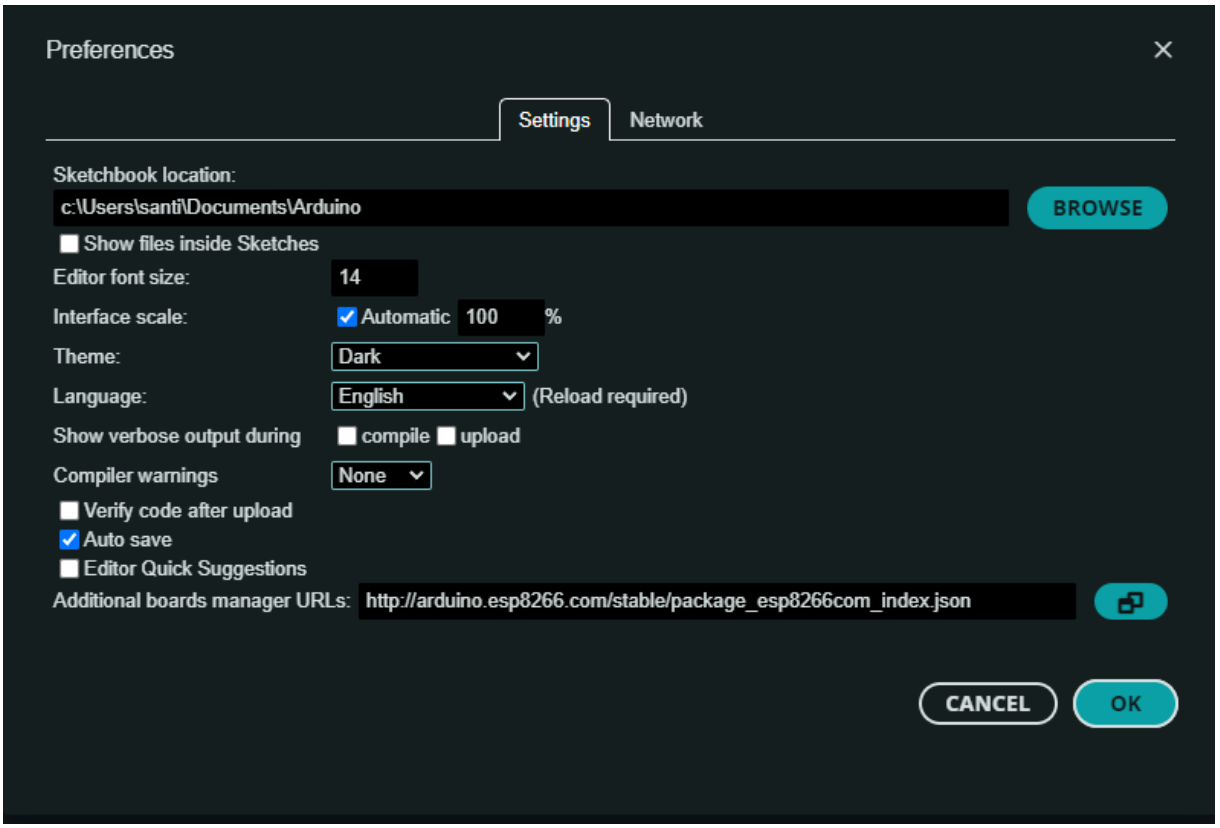
### 4.3 Código Microcontrolador

O código do microcontrolador tem como objetivo estabelecer a conexão Wi-Fi, possibilitando a comunicação entre o dispositivo e o aplicativo via *broker* MQTT. Além disso, ele gerencia o controle da válvula solenóide, realiza a leitura e envio de informações, como a umidade do solo, e armazena configurações definidas pelo usuário no aplicativo.

### 4.3.1 Ambiente e Bibliotecas Utilizadas

Para o desenvolvimento do código, foi utilizada a Arduino IDE versão 2.3.4, configurada para compilar e carregar o programa no ESP8266. A configuração é feita adicionando o *link* correspondente no campo "URLs Adicionais para Gerenciadores de Placas", conforme ilustrado na Figura 15.

Figura 15 - Página preferências arduino IDE com placa ESP8266 inserida




Fonte: Acervo pessoal (2025).

Além disso, é necessário instalar o driver CH340G para que o computador reconheça o microcontrolador conectado via USB, permitindo a transferência do código para a placa.

Com o ambiente devidamente configurado, algumas bibliotecas foram instaladas para fornecer funcionalidades essenciais:

- *ESP8266WiFi.h*: Gerencia a conexão Wi-Fi do ESP8266, incluindo conexão a redes, criação de pontos de acesso e atribuição de IPs (*Internet Protocol*).

- *WiFiManager.h*: Trabalha em conjunto com a biblioteca anterior, gerenciando conexões Wi-Fi automaticamente sem a necessidade de reprogramação, criando um ponto de acesso acessível por dispositivos móveis para configuração da rede.
- *EEPROM.h*: Permite a leitura e gravação de dados na memória EEPROM (*Electrically Erasable Programmable Read-Only Memory*) do ESP8266, garantindo que configurações permaneçam salvas mesmo após o reinício do dispositivo.
- *PubSubClient.h*: Implementa o protocolo MQTT, possibilitando a conexão com *brokers*, além de publicação e assinatura de tópicos.
- *Arduino_JSON.h*: Facilita a manipulação de objetos JSON. Essa biblioteca permite a criação, análise e modificação de mensagens JSON, simplificando a troca de dados entre o ESP8266 e o aplicativo móvel.
- *NTPClient.h*: Sincroniza o relógio do microcontrolador com servidores NTP (*Network Time Protocol*), eliminando a necessidade de componentes físicos como RTCs (*Real Time Clock)* para manter informações precisas de data e hora.

### 4.3.2 Configuração Wi-Fi

A configuração do Wi-Fi ocorre da seguinte forma: inicialmente, caso não haja nenhuma rede previamente salva na memória EEPROM do microcontrolador, o ESP8266 cria um ponto de acesso próprio, permitindo que um dispositivo móvel, como um *smartphone*, se conecte via Wi-Fi.

Ao estabelecer essa conexão, o ESP abre automaticamente uma interface no navegador do dispositivo, exibindo uma lista com todas as redes Wi-Fi disponíveis. O usuário pode então selecionar uma rede, inserir a senha correspondente e definir um nome para o dispositivo. Esse nome é arbitrário, mas deve ser o mesmo utilizado no aplicativo para garantir a sincronização.

Após a configuração, o ESP tenta se conectar à rede selecionada. Em caso de sucesso, as credenciais são armazenadas na EEPROM para uso futuro. Assim, sempre que o ESP for reiniciado, ele tentará se conectar automaticamente à última rede salva.

### 4.3.3 Conexão *Broker* MQTT

Para conexão do *broker* MQTT, é criado um *client* utilizando a biblioteca *PubSubClient.h*. Além disso, o servidor foi definido como *broker.hivemq.com* e a porta como 1883, que é uma porta TCP (*Transmission Control Protocol*) não segura, devido ao *broker* ser gratuito.

Após a configuração do *broker*, define-se uma função de *callback* que será acionada sempre que o ESP receber uma mensagem de um tópico ao qual esteja inscrito. Em seguida, inicia-se o processo de conexão e, caso a tentativa seja bem-sucedida, o dispositivo se inscreve nos tópicos necessários para a comunicação.

Com o *broker* já conectado, é realizada a inscrição em todos os tópicos relevantes para o funcionamento do sistema e além disso é enviada uma mensagem para o tópico principal com o nome do dispositivo indicando a conexão do mesmo. Se a conexão falhar, o sistema aguarda um segundo antes de tentar novamente.

### 4.3.4 Recebimento de Mensagens

As mensagens recebidas são processadas pela função de *callback* configurada anteriormente. Essa função recebe o nome do tópico, um ponteiro para o conteúdo da mensagem e seu comprimento. A mensagem é então armazenada em uma *string* para facilitar o processamento.

Após essa separação, é realizada uma verificação se a mensagem é proveniente do aplicativo ou do controlador. Com essa verificação, é realizada uma série de condicionais para verificar para qual finalidade essa mensagem foi enviada e tratar ela do jeito correto.

Caso seja um pedido de conexão do aplicativo, o ESP define uma variável interna indicando que a comunicação foi estabelecida e envia uma mensagem de confirmação ao aplicativo. Mas caso a mensagem recebida seja a confirmação da conexão, é apenas atualizado o *status* de conexão, garantindo que ambos os dispositivos estejam sincronizados.

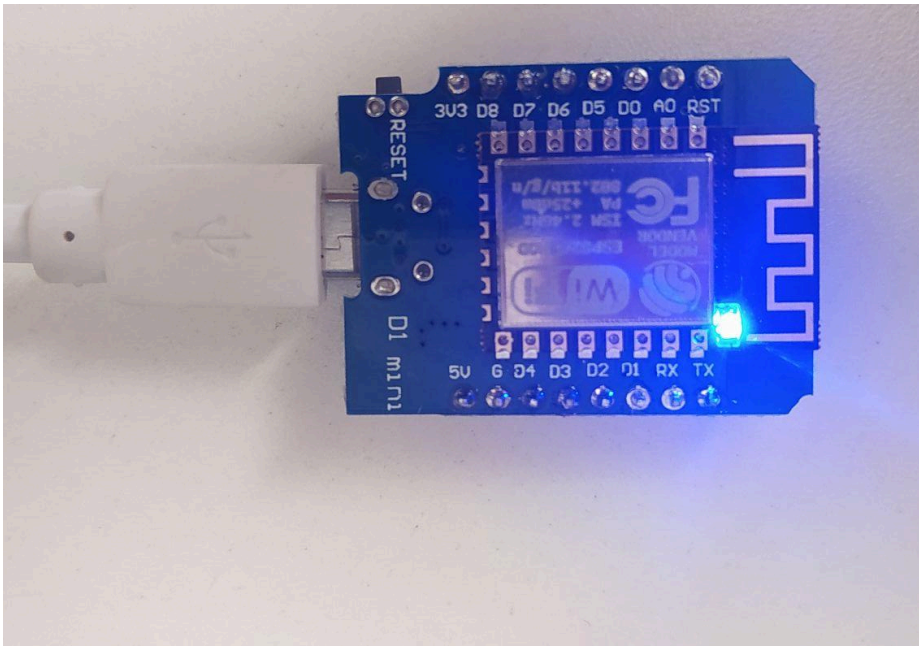
Os próximos condicionais verificam qual o tópico da mensagem, desestruturando o tópico e obtendo o terceiro nível dele. Com isso é possível identificar qual a finalidade do envio da mensagem e tratá-la corretamente. Então de acordo com o tópico, é executada uma função específica que realizará uma ação de acordo com o conteúdo da mensagem.

Nota-se que uma alternativa para a utilização dos condicionais seria o mapeamento de funções dinâmicas, como por exemplo *hash tables*, sendo mais modular e eficiente que o uso de um grande número de condicionais.

### 4.3.5 Estados Conexão

O ESP 8266 D1 Mini possui um LED interno conectado ao GPIO (*General Purpose Input/Output*) de número dois, como demonstrado na Figura 16.

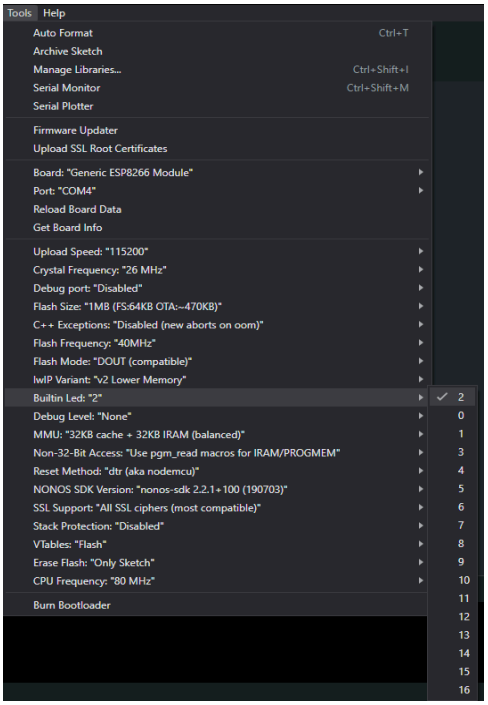
Figura 16 - LED interno ESP 8266 D1 Mini




Fonte: Acervo pessoal (2025).

Esse LED pode ser utilizado quando configurado o parâmetro *Builtin Led* dentro da IDE Arduino, como demonstrado na Figura 17.

Figura 17 - Configuração do *Builtin LED* dentro da IDE Arduino




Fonte: Acervo pessoal (2025).

O LED interno do ESP8266 opera em lógica invertida (ativo-baixo), ou seja, acende quando o pino GPIO2 está em nível lógico baixo e apaga quando está em nível alto.

Dessa forma, foi definido três estados para a primeira conexão com o aplicativo móvel. O primeiro estado se refere no qual o dispositivo está desconectado do *broker* e, consequentemente, da rede Wi-Fi. O segundo, a unidade de controle está conectada corretamente ao *broker* mas ainda não obteve informação de conexão do aplicativo. E por fim, o último estágio se refere a possibilidade de comunicação entre o dispositivo físico e o aplicativo móvel por meio do *broker* MQTT.

O primeiro estágio se ativa assim que o ESP é conectado à energia. Consiste em manter seu LED embutido ligado, exibindo assim que não houve conexão com o *broker*, sendo necessário configurar inicialmente o Wi-Fi ou então, deletar as informações da rede antiga e conectar em alguma outra que o dispositivo esteja reconhecendo.

Já no segundo, o LED alterna entre ligado e desligado a cada segundo, indicando que o ESP já está no *broker*, mas ainda aguarda o contato do aplicativo. Esse problema pode ocorrer caso o *smartphone* esteja sem *internet* ou tenha falhado na conexão ao *broker*.

Por fim, o LED desligado indica o terceiro estágio, no qual a conexão foi concluída e o dispositivo já conseguiu se comunicar com o aplicativo móvel. Foi decidido que nessa etapa o LED estaria desligado, visto que idealmente essa é a etapa que o dispositivo ficará quando

estiver em seu completo funcionamento. Dessa forma, desligando o LED poupa energia, permitindo que o sistema se mantenha por mais tempo funcionando.

Esse ciclo de estados ocorre apenas na primeira conexão. Após a configuração inicial, o ESP é instalado na plantação e não há necessidade de monitoramento visual constante. O objetivo do projeto é permitir o gerenciamento remoto da irrigação, tornando desnecessário repetir esse ciclo sempre que o celular do usuário se desconectar temporariamente.

### 4.3.6 Reiniciar Dispositivo e Esquecer Rede

O circuito físico conta com dois botões para ações manuais. O primeiro deles é responsável por reiniciar o sistema manualmente. Já o segundo permite que sejam apagadas as credenciais Wi-Fi salvas, permitindo a conexão de uma nova rede.

Embora essas funções também possam ser acionadas via aplicativo, os botões físicos são úteis em situações onde o acesso ao aplicativo não é possível, como em casos de troca de senha do roteador. Sem essa opção, o dispositivo ficaria inacessível até que um novo comando fosse enviado remotamente, mas com essa opção, é possível limpar os dados diretamente no dispositivo.

A verificação do estado dos botões ocorre a cada *loop* de execução do código. Dessa forma, basta pressioná-los por um segundo até que o microcontrolador leia a mudança do nível lógico de suas portas GPIO e execute a função.

Além disso, é possível executar essas duas funções remotamente pelo aplicativo. O subtópico *Config* é utilizado para a ativação dessas funções. Caso o aplicativo envie uma mensagem "*Reload"*, ocorrerá o reiniciamento do dispositivo, mas caso a mensagem for "Forget", a rede armazenada na memória será apagada, do mesmo modo que ocorre quando pressionados os botões.

### 4.3.7 Envio da Umidade

O envio do valor da umidade é realizado por uma função que é executada no início de cada *loop* de execução. Consiste de realizar a leitura do sensor analógico de umidade, que recebe valores de 0 a 1024 de acordo com a umidade, sendo que quanto menor o valor, maior a umidade do solo.

Após a obtenção da leitura, o valor é convertido para uma escala percentual, facilitando sua interpretação. Se o ESP estiver conectado ao *broker*, os dados de umidade são enviados para o tópico correspondente, permitindo que o aplicativo exiba essas informações ao usuário.

### 4.3.8 Configuração de Parâmetros para Irrigação

O subtópico *IrrigateParams* é utilizado para configuração de parâmetros relacionados à duração da irrigação ou ao percentual de umidade do solo necessário para a ativação do sistema. Dessa forma, o sistema extrai as informações e as armazena em variáveis globais, indicando o tipo e valores da irrigação definida pelo usuário.

Esses valores são armazenados em variáveis globais que são utilizados tanto na função de ativação instantânea quanto na ativação programada por dia. Eles servem como limites para que o sistema determine o momento adequado para encerrar a irrigação.

### 4.3.9 Função de Programar o Dispositivo

Quando é recebida uma mensagem pelo subtópico *Program*, o sistema interpreta como uma nova configuração de programação. A mensagem recebida é desestruturada, e seus dados são armazenados para posterior utilização. Sua estrutura é constituída de uma *string* com números separados pelo caractere “-”, permitindo que os dados sejam interpretados corretamente.

A primeira posição indica se condição de irrigação por dia da semana foi ativada, a segunda a condição por horário e a terceira, a de umidade. Já a quarta posição contém os dias da semana selecionados, representados por números de 0 (domingo) a 6 (sábado). A quinta posição indica o horário de início e o final concatenados no formato HH:MM. E por fim, a última posição indica o nível de umidade do solo necessário para iniciar a irrigação.

### 4.3.10 Ativação e Desativação do Sistema

A ativação manual pode ser realizada a qualquer momento, desde que seja enviada uma mensagem para o subtópico *Activate* requisitando a ativação. Dessa forma, será realizada a ativação da válvula solenóide, liberando a água para a irrigação, até que a condição escolhida pelo usuário seja satisfeita.

No caso da ativação manual, a cada ciclo de execução, o sistema verifica se o tempo decorrido desde a ativação ultrapassou o limite configurado ou se a umidade do solo atingiu o valor predefinido. Se uma dessas condições for satisfeita, a irrigação é interrompida.

Para a irrigação programada, o sistema avalia se alguma condição estabelecida pelo usuário está ativa. O comportamento do sistema varia conforme a combinação de condições programadas, conforme ilustrado no Quadro 1.

Quadro 1 - Combinação de condições de seleção e comportamento esperado

| **Combinação** | **Comportamento** |
|---|---|
| Dia da semana | Irriga conforme o tempo ou a umidade configurada, apenas nos dias selecionados. |
| Horário | Irriga diariamente dentro do horário especificado. |
| Umidade | Irriga em qualquer momento, sempre que a umidade estiver abaixo do limite definido |
| Dia da semana + Horário | A irrigação ocorre apenas nos dias programados, dentro do intervalo de horário definido. |
| Dia da semana + Umidade | Nos dias programados, a irrigação é ativada quando a umidade fica abaixo do valor especificado. |
| Horário + Umidade | Em todos os dias, a irrigação é ativada dentro do horário especificado caso a umidade esteja abaixo do limite. |
| Dia da semana + Horário + Umidade | Nos dias programados, a irrigação ocorre dentro do horário definido e somente se a umidade estiver abaixo do limite. |

Fonte: Autoria própria (2025).

Caso apenas a condição do dia da semana seja selecionada, a irrigação ocorre conforme o tempo ou a umidade configurados, funcionando de maneira similar à ativação manual. Além disso, o sistema registra que a irrigação já ocorreu naquele dia, impedindo a repetição indesejada ao longo do mesmo dia. Essa condição é redefinida automaticamente à meia-noite.

Dessa forma, se por exemplo, fosse selecionado apenas o dia da semana, ocorreria uma irrigação por dia de acordo com os parâmetros de tempo ou irrigação pré-estabelecidos. Mas caso selecionasse também um horário, o sistema seria ativado durante toda a janela de horário

naquele dia da semana selecionado, não levando em consideração a configuração válida para apenas o dia da semana.

#### 4.3.11 Gerenciamento do Tempo

No início de cada ciclo de execução, o sistema executa uma função específica para o gerenciamento do tempo do dispositivo. Essa função atualiza a variável de tempo real por meio do protocolo NTP, extraindo e formatando as informações de horas e minutos para utilização posterior.

Além disso, é verificado se o horário atual é exatamente meia-noite. Caso positivo, o sistema redefine a condição que impede irrigação duplicada dentro do mesmo dia, garantindo que a irrigação programada possa ser executada normalmente no dia seguinte.

Por fim, após a atualização do horário, o valor é repassado à função responsável pela verificação das condições de irrigação, assegurando que todas as decisões do sistema sejam baseadas em um horário sempre atualizado.

### 4.4 Aplicativo IrrigaHub

Para possibilitar a comunicação entre o dispositivo e o usuário, foi desenvolvido um aplicativo móvel inicialmente disponibilizado para dispositivos com sistema operacional Android.

O aplicativo foi desenvolvido utilizando o *framework* React Native em conjunto com a ferramenta Expo, que facilita a criação de aplicativos ao fornecer um ambiente otimizado para desenvolvimento e testes. O React Native permite a criação de aplicativos móveis utilizando a biblioteca JavaScript React, baseada na reutilização de componentes independentes, garantindo um código mais organizado e otimizado.

Assim, com a utilização do React Native é possível desenvolver um código único para IOS e Android utilizando JavaScript. Aliando-o ao Expo, é possível realizar o desenvolvimento de aplicativos sem a necessidade de um ambiente nativo, dessa forma, é possível testar diretamente o aplicativo no próprio celular enquanto há o desenvolvimento, além de facilitar a *build* e geração do arquivo *.apk*, arquivo esse utilizado para instalação do aplicativo em celulares Android.

Além disso, é válido ressaltar que o aplicativo não faz uso de um *backend* ou banco de dados, pois, inicialmente, será utilizado por poucos usuários. Dessa forma, elimina-se a

necessidade de um servidor pago ou gratuito com funcionamento instável. Assim, todas as informações são armazenadas no próprio dispositivo do usuário e a comunicação ocorre exclusivamente via protocolo MQTT.

### 4.4.1 Configuração da Plataforma

Inicialmente para que seja possível executar um projeto React Native, é necessário instalar o Node.js no computador do desenvolvedor. Esse *software* permite a execução de códigos JavaScript fora de um navegador *web*.

Após isso, utilizando o gerenciador de pacotes do Node.js, o NPM (*Node Package Manager*), é possível realizar a instalação do Expo, plataforma que facilita o desenvolvimento e configuração inicial de projetos em React Native. O Expo realiza as pré-configurações necessárias e permite a geração do arquivo final para *download* em dispositivos móveis.

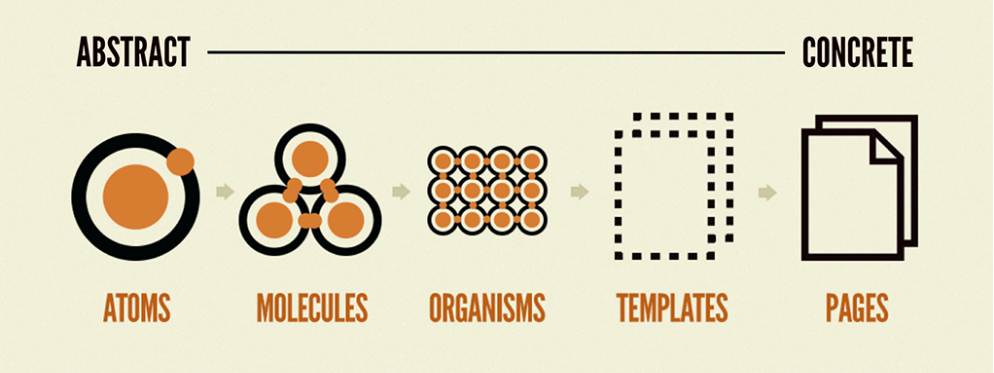
Além disso, foi utilizada a plataforma de edição de texto Visual Studio Code que possui diversas ferramentas para edição de código, além de extensões que aumentam a eficiência durante o desenvolvimento do código, como atalhos e preenchimento automático.

### 4.4.2 Estruturação do Aplicativo

A estrutura do aplicativo é organizada em uma pasta raiz contendo diretórios específicos para componentes, configurações, contextos, telas e outras funcionalidades.

Os componentes do aplicativo foram baseados na metodologia *Atomic Design*, criada por Brad Frost (FROST, 2016). Ela divide a interface em cinco níveis hierárquicos inspirados na química: átomos, moléculas, organismos, templates e páginas. Isso facilita a reutilização e manutenção dos componentes. Essa organização é exemplificada pela Figura 18.

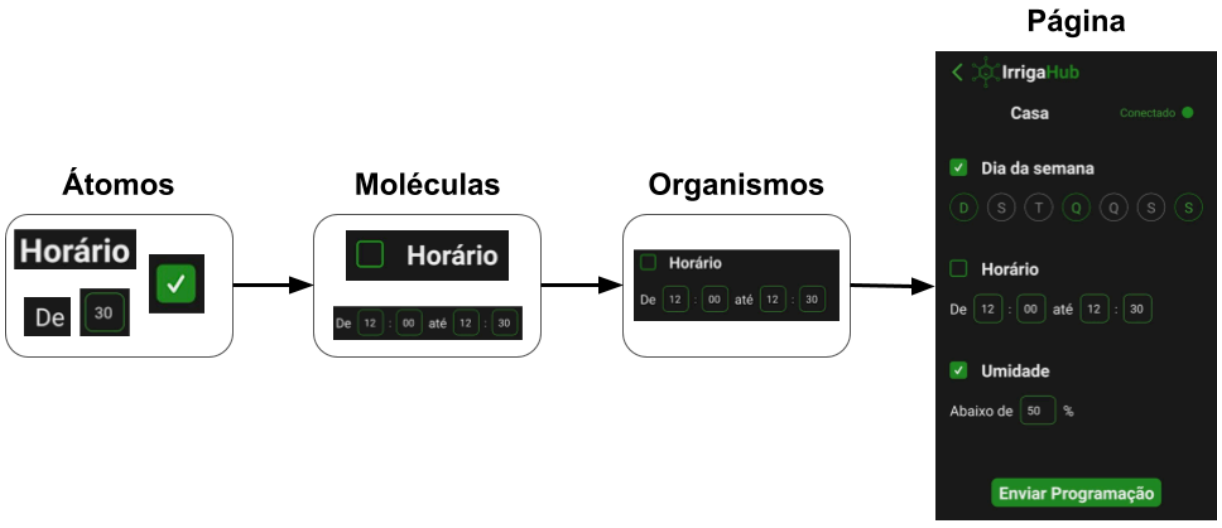
Figura 18 - Representação do Atomic Design




Fonte: Brad Frost (2016).

No projeto, a estrutura foi simplificada para três níveis: átomos, moléculas e organismos, devido ao seu tamanho reduzido. Em suma, as páginas do aplicativo são compostas por organismos, que, por sua vez, são formados por moléculas e, no nível mais básico, por átomos. Dessa forma, alterações em um componente atômico refletem automaticamente em moléculas e organismos em todo o aplicativo. Tal utilização pode ser visualizada na Figura 19.

Figura 19 - Utilização do Atomic Design no projeto




Fonte: Autoria própria (2025).

A pasta de configuração contém constantes globais, como cores, tamanhos e espessura das fontes, garantindo padronização e facilitando a manutenção do design. Outras pastas armazenam arquivos responsáveis pela comunicação MQTT, contextos, armazenamento de dados e telas do aplicativo.

### 4.4.3 Comunicação MQTT

Para a comunicação MQTT foi utilizada a biblioteca Paho MQTT, uma biblioteca capaz de realizar a comunicação desse protocolo integrado com o React Native. Para gerenciar essa conexão, foi criado um arquivo específico que centraliza todas as funções relacionadas ao MQTT.

Inicialmente, um cliente é criado e conectado ao *broker* MQTT, utilizando credenciais apropriadas. Esse cliente gerencia as conexões e desconexões dos tópicos, permitindo a inscrição para recebimento de mensagens e a ativação de *callbacks* para processá-las. Também há uma função dedicada ao envio de mensagens para tópicos específicos.

### 4.4.4 Gerenciamento de Contexto

O gerenciamento de estados globais é feito por meio do Context API (*Application Programming Interface*) do React, permitindo que informações como status de conexão e mensagens recebidas sejam acessadas independentemente da tela exibida no momento.

Em um primeiro momento, essa API foi escolhida devido a não utilização de *back-end*, simplificando a passagem de poucos dados. Porém, para grandes quantidades de dados ele fica ineficiente. Assim, para manipulação de diversos dados pode-se utilizar um *back-end* centralizado que faria a gestão das mensagens, não necessitando do Context API.

O contexto armazena todas as mensagens recebidas e os tópicos inscritos, permitindo o acesso aos dados mediante conhecimento do tópico correspondente. Além disso, a inscrição e desinscrição de tópicos são gerenciadas para evitar processamento desnecessário de mensagens.

Assim, qualquer estado de qualquer tópico pode ser tratado paralelamente às páginas, evitando que em cada página acessada, seja necessário realizar novamente a inscrição nos mesmos tópicos já inscritos.

### 4.4.5 Armazenamento de Dados

O aplicativo não utiliza *backend* ou banco de dados, foi utilizada uma biblioteca chamada *LocalStorage*. Essa biblioteca, não nativa do React Native, sendo instalada via NPM (BEVACQUA, 2019), permite o armazenamento de informações no cache do aplicativo, permitindo assim a substituição de um servidor de banco de dados pelo próprio dispositivo do usuário, visto que o aplicativo será inicialmente utilizado por poucas pessoas, não havendo a necessidade de um servidor central com informações.

Dessa forma, há um arquivo dedicado a armazenamento de informações. Possui funções de salvar itens, carregar itens e remover itens da memória que não são utilizados. Essas funções podem ser exportadas para qualquer parte do aplicativo, possibilitando seu uso independente da tela renderizada.

### 4.4.6 Tela Inicial

A tela inicial é o primeiro contato que o usuário terá com o sistema como um todo. Nela haverá a tentativa de conexão com o *broker* MQTT que, no caso de sucesso, exibe a possibilidade de adicionar novos dispositivos.

Quando adicionado um novo dispositivo, haverá uma tentativa inicial de conexão com esse dispositivo em específico. Desde que o dispositivo tenha o mesmo nome daquele especificado na unidade de controle e, ambos estejam conectados ao *broker*, haverá a mudança do estado do dispositivo selecionado para “Conectado”. Além disso, caso seja do desejo do usuário, é possível pressionar o botão que indica a reconexão, reiniciando o estado de conexão de todos os dispositivos listados.

Nesta tela ainda é possível adicionar vários outros dispositivos, desde que possuam nomes diferentes, permitindo realizar o controle e monitoramento de várias unidades de controle diferentes. Quando pressionado o nome de um desses dispositivos, haverá a mudança de tela, exibindo a tela de gerenciamento do dispositivo.

### 4.4.7 Tela de Gerenciamento

Esta tela permite que o usuário realize diversas configurações e visualize informações referente ao dispositivo escolhido.

No cabeçalho da tela é possível observar as opções de configurações e exclusão. Quando pressionada a engrenagem, abrirá um menu que permite o usuário escolher remotamente entre reiniciar o dispositivo físico ou então deletar a rede registrada, controlando diretamente o ESP. Além disso, no botão de lixeira, o usuário pode excluir esse dispositivo, retornando então automaticamente para a tela inicial, permitindo que seja escolhido um dispositivo diferente, ou então acrescentado um novo.

Também são exibidos o nome do dispositivo e o estado de conexão. Esse estado é regido de acordo com o recebimento de mensagens, visto que sempre que aberto essa tela, o aplicativo se inscreve no tópico referente às informações de umidade. Caso o aplicativo não receba a atualização da umidade após um tempo definido, é realizada a mudança de estado para “Desconectado”. Mas caso seja recebida uma nova mensagem, o contador é reiniciado e então o estado é definido como “Conectado”.

A umidade recebida pelo tópico é então atualizada em tempo real de acordo com o envio pelo dispositivo físico. Essa informação é exibida para o usuário na parte inferior da tela. Há dois botões nessa tela, sendo eles o primeiro que leva para a próxima tela, a de programação, e o outro que ativa o sistema de irrigação com o seu toque. Além disso, há uma barra deslizante que permite realizar a escolha do valor dos parâmetros para a irrigação.

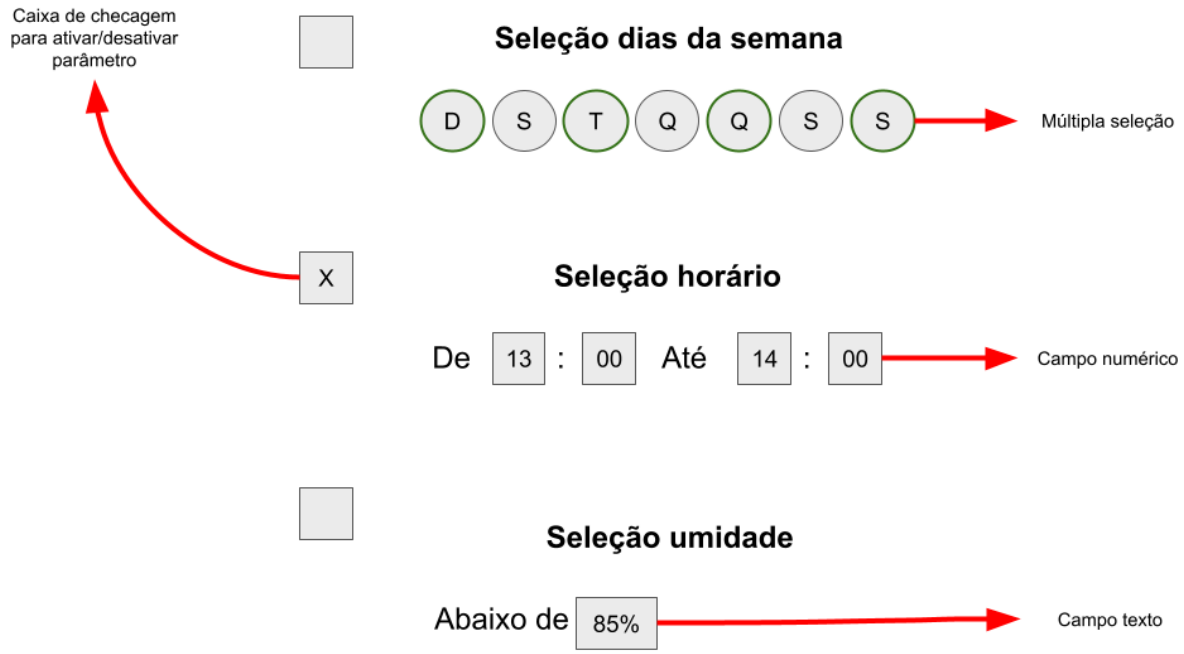
É possível alternar entre tempo de irrigação, que vai até uma hora, possuindo a divisão de um minuto, ou então a umidade do solo que o sistema deve parar de irrigar, indo até cem por cento.

Com esses parâmetros escolhidos, quando se pressiona o botão “Ativar”, é realizada uma verificação se o sistema está conectado, caso contrário uma mensagem de erro é exibida. Por outro lado, se o sistema estiver conectado, é enviada então a mensagem com a formatação correta para que o sistema físico irrigue naquele instante até a condição enviada. Essas informações são armazenadas no *cache* do celular, permitindo que fiquem salvas mesmo após fechar o aplicativo.

### 4.4.8 Tela de Programação

Por fim, o botão de ativar leva à tela de programação, que permite realizar a programação do dispositivo para que o mesmo atue automaticamente. As opções são: ativação por dia da semana, horário ou umidade, como observado na Figura 20.

Figura 20 - Projeto tela de programação do aplicativo IrrigaHub




Fonte: Autoria própria (2025).

Cada opção possui uma caixa de seleção ao lado do título. A seleção desta caixa indica se a informação será considerada ou ignorada pelo dispositivo físico quando realizar a checagem de condição de ativação.

É possível selecionar os dias da semana de acordo com a inicial de cada dia. Elas são dispostas em linha, havendo uma mudança de cor ao pressionar, indicando que aquele dia está selecionado. No fim é enviado ao ESP um concatenado com os números que indicam cada dia da semana.

Também há a seleção por horário, na qual o usuário deve escrever as horas e minutos do horário inicial e final. É aberto um teclado numérico para o usuário imputar a informação necessária em cada campo, além de uma verificação que impede a utilização de horários inexistentes.

A terceira opção é a umidade, que consiste em um *input* numérico que aceita valores positivos inteiros menores ou iguais a cem. Caso seja colocado um valor que foge desse intervalo, é automaticamente convertido ao valor mais próximo, seja ele zero ou cem.

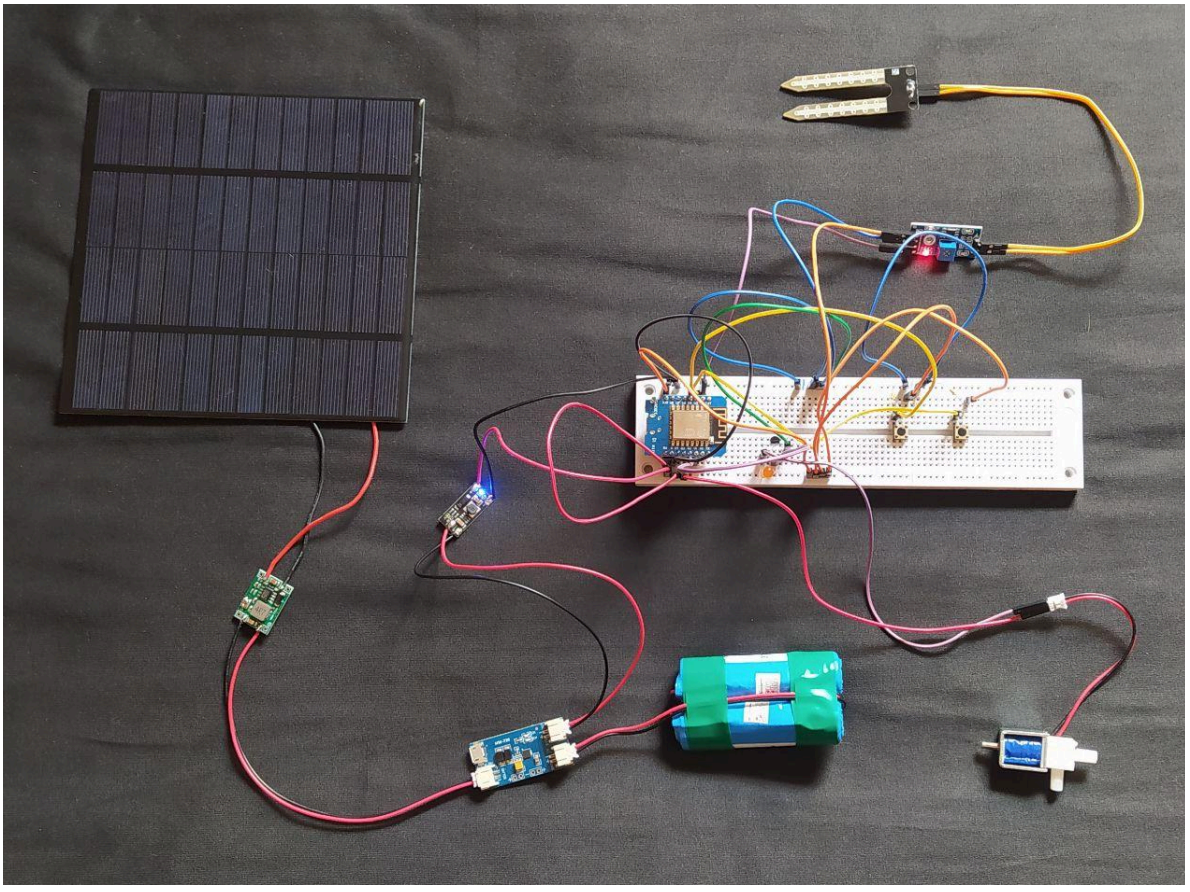
Por fim, basta pressionar o botão localizado no final da tela para realizar o envio das informações ao dispositivo físico. Do mesmo jeito da tela anterior, também é exibido o estado de conexão do dispositivo, sendo utilizado para exibir ao usuário se as informações foram enviadas ou não.

# 5 RESULTADOS E DISCUSSÃO

Este projeto tem como objetivo permitir que o usuário monitore, em tempo real, a umidade do solo de sua cultura, além de possibilitar a ativação remota do sistema de irrigação, seja de forma instantânea por comando ou automaticamente conforme uma programação prévia. Além disso, buscou-se desenvolver um sistema completamente *off-grid*, ou seja, independente da rede elétrica, utilizando a estratégia de *Energy Harvesting* por meio de um painel solar.

Inicialmente, como se trata de um protótipo, o circuito foi montado em uma *protoboard*, permitindo modificações rápidas ao longo do desenvolvimento. O protótipo final pode ser observado na Figura 21.

Figura 21 - Protótipo final físico



Fonte: Acervo pessoal (2025).

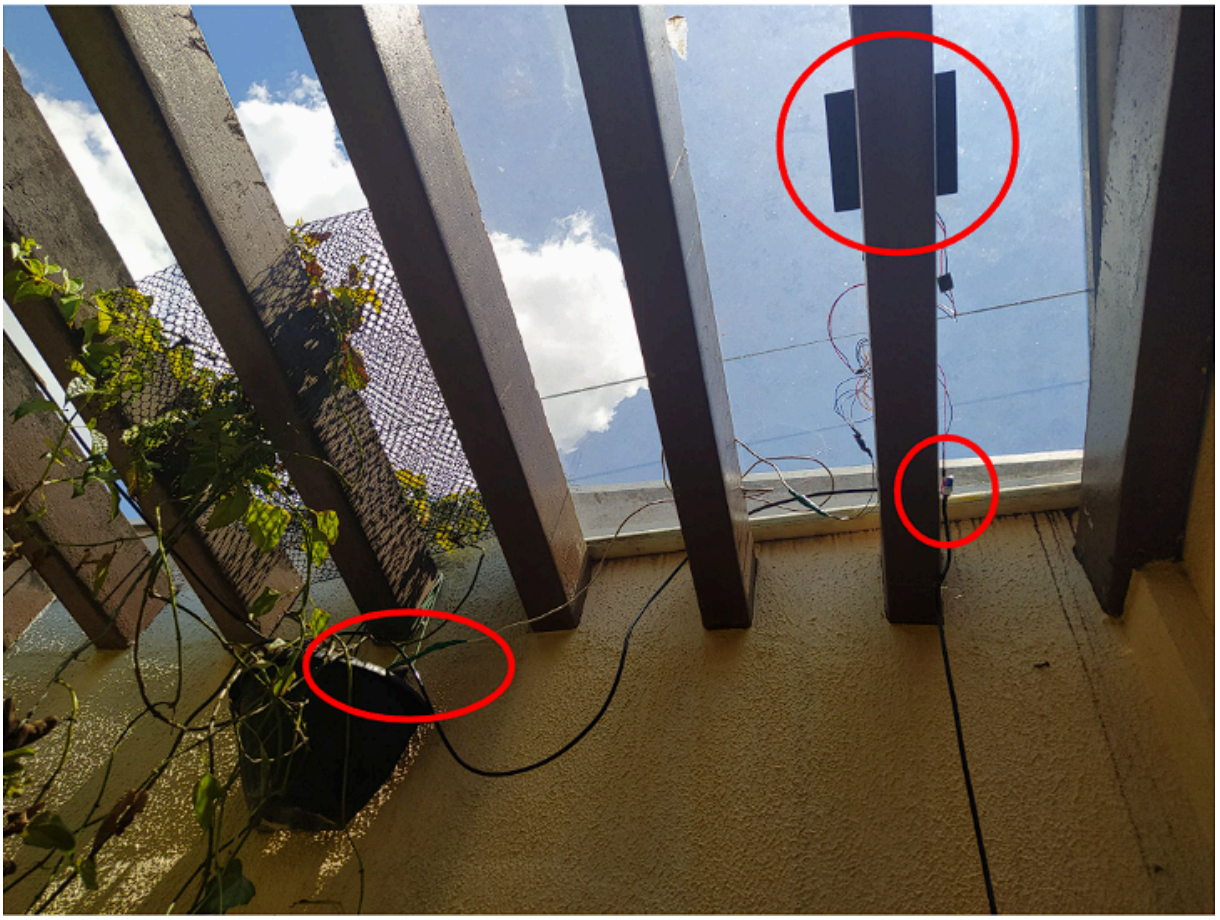
Para realizar os testes finais, foi utilizado um jardim de inverno com dimensões aproximadas de 1,30 x 2,80 x 3,00 m, cujas culturas estão dispostas em diferentes posições, como observado na Figura 22.

Figura 22 - Local de instalação do sistema

Fonte: Acervo pessoal (2025).

O ambiente já possuía um sistema de irrigação baseado na pressão da rede de abastecimento residencial, sendo suficiente para irrigar todas as plantas ao abrir a torneira. Dessa forma, é possível realizar a instalação do sistema de irrigação em um local próximo a uma das culturas e do duto, permitindo a atuação da válvula solenóide, como demonstrado na Figura 23.

Figura 23 - Instalação do protótipo com destaque no sensor de umidade, placa solar e válvula solenóide, da esquerda para a direita.



Fonte: Acervo pessoal (2025).

Na Figura 23, observa-se um duto plástico preto, responsável por conduzir a água da torneira até as plantas, passando pela válvula solenóide (indicada pelo círculo vermelho no centro-direito da imagem). Também é possível identificar a placa solar, posicionada estrategicamente para captação de luz, e o sensor de umidade, inserido no solo do vaso destacado à esquerda. O local de instalação conta ainda com um vidro protetor, que resguarda os componentes eletrônicos contra intempéries como chuva e poeira. Com o sistema instalado, foi possível a realização de um total de três testes.

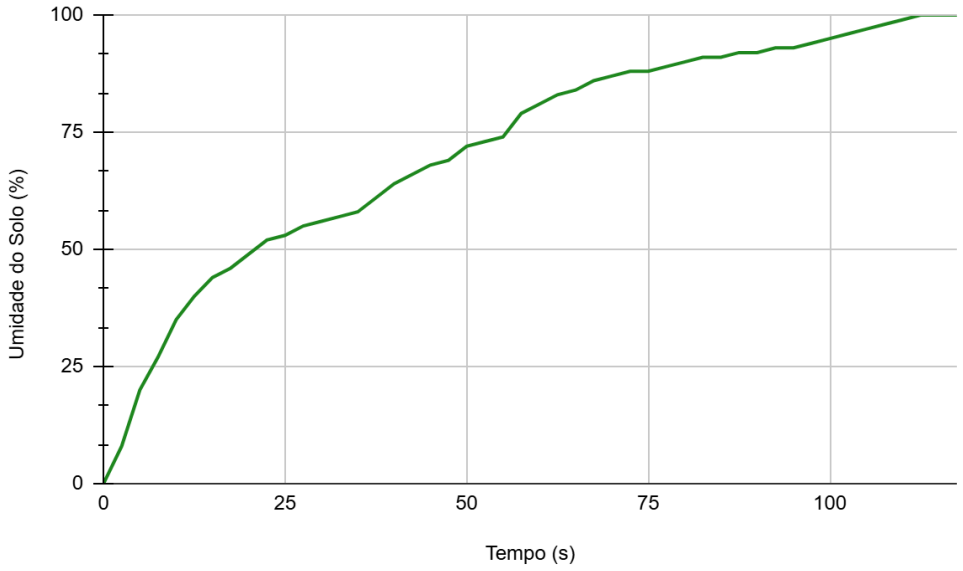
O primeiro teve como objetivo avaliar a autonomia total do protótipo utilizando exclusivamente a bateria e alimentação solar. Esse teste foi dividido em duas etapas: na primeira, a placa solar foi desconectada para mensurar a autonomia da bateria; na segunda, a bateria pôde ser recarregada pelo painel solar. É importante destacar que, durante esse teste, o sistema foi programado para realizar uma irrigação diária de 15 minutos.

Com esse primeiro teste é possível constatar que o sistema projetado possui independência da rede elétrica, garantindo aproximadamente 36 horas de autonomia sem luz solar quando a bateria está totalmente carregada. Já quando exposto à luz solar por aproximadamente 12 horas diárias, a autonomia pode chegar a 55 horas. Nota-se que a autonomia foi menor que a prevista na especificação da bateria (65 horas), devido às perdas dos conversores e outros componentes do circuito.

No entanto, não foi possível garantir o funcionamento exclusivamente com energia solar. Apesar da placa solar fornecer potência suficiente durante os períodos de maior incidência solar, a carga gerada ao longo do dia não foi suficiente para suprir o consumo total do sistema. Como resultado, a bateria se descarregou progressivamente ao longo do dia, mesmo em período com luz solar presente, impedindo que o sistema opere 100% *off-grid*.

Já o segundo teste foi em relação ao sensor de umidade e velocidade de irrigação do sistema instalado. Foi utilizada uma terra totalmente seca e um vaso para realizar esse teste, sendo que o sensor, quando instalado, registrou 0% de umidade no solo. Além disso, foi utilizado o próprio sistema de irrigação instalado para irrigar essa porção de terra. Como resultado, pôde-se obter o gráfico exibido na Figura 24 exibindo a porcentagem de umidade do solo desde o início da irrigação até o valor de 100% em função do tempo.

Figura 24 - Representação gráfica umidade do solo em relação ao tempo de irrigação com a válvula ativada




Fonte: Autoria própria.

Dessa forma, observa-se que o sensor de umidade apresentou boa precisão na medição da umidade relativa do solo, enquanto a válvula solenóide proporcionou vazão suficiente para a irrigação utilizando apenas a pressão da rede de abastecimento, eliminando a necessidade de uma bomba adicional. A referência da umidade foi estipulada manualmente, sendo 0% um solo totalmente seco, correspondendo a 2,51 V, e 100% um solo encharcado correspondendo a 1,03 V.

Além disso, verificou-se que, partindo de um solo completamente seco, é possível deixá-lo úmido em aproximadamente 2 minutos de irrigação. Nota-se que em condições reais de uso, quando o solo dificilmente estará totalmente seco, o tempo necessário para irrigação será menor que o tempo registrado.

Outro ponto relevante é que, inicialmente, há uma variação percentual mais acentuada no aumento da umidade quando o solo está seco, que diminui à medida que o solo passa de 50% de umidade. Isso demonstra que o sensor tem maior sensibilidade para detectar variações entre o solo seco e moderadamente úmido, com uma redução dessa sensibilidade conforme a umidade aumenta além desse ponto.

Um último teste realizado simula a utilização real do sistema, monitorando a umidade durante um período de aproximadamente duas horas. Para esse teste, o sistema foi programado para garantir que a umidade do solo, partindo de aproximadamente 100% de umidade, não fosse menor que 85%, fazendo com que a válvula seja acionada quando necessário, com aquisição dos dados a cada 10 segundos. Além disso, foram colocadas duas lâmpadas halógenas de 70 W com o objetivo de fazer o solo secar mais rapidamente, permitindo a realização do teste em um tempo mais curto. O resultado pode ser observado na Figura 25.

Figura 25 - Representação gráfica umidade do solo em relação ao tempo de irrigação com umidade programada


Fonte: Autoria própria.

Observa-se que o sistema manteve a umidade do solo, na maior parte do tempo, acima do valor programado, ativando a válvula automaticamente quando a umidade ficou abaixo do limite estabelecido, por volta das 17:10. Isso comprova que o sistema funcionou conforme o esperado, garantindo que a umidade não ficasse inferior ao valor requisitado.

No entanto, é possível notar uma diminuição brusca do valor lido de aproximadamente 10 pontos percentuais ao acionar a válvula solenóide, esse comportamento se deve a imprecisões do sensor que ocorrem devido ao elevado consumo de corrente da válvula, que demanda até três vezes mais energia do que o restante do sistema em operação normal.

Essa sobrecarga provoca uma queda temporária na tensão fornecida pela bateria, o que afeta o desempenho dos demais componentes, especialmente o sensor de umidade. Durante o acionamento da válvula, o sensor registra valores inferiores aos reais, levando o sistema a interpretar que o solo ainda está seco e, consequentemente, a manter a irrigação por mais tempo.

Quando a válvula é finalmente desligada, a tensão se estabiliza, permitindo que o sensor retome leituras precisas, sendo possível observar no gráfico grandes variações da medição. Por conta disso, a umidade do solo não se estabiliza exatamente em 85%, pode-se perceber que, ocorre um aumento de aproximadamente 15 pontos percentuais do valor estabelecido.

No momento em que o sensor retoma a leitura correta quando a válvula se desliga, há uma elevação de aproximadamente 10% de umidade, compensando a imprecisão que o sensor demonstra quando a válvula está acionada. Os outros 5% se dão ao atraso proveniente da água presente na tubulação que, mesmo com a válvula desligada, continua a circular por ação da gravidade, elevando ainda mais o valor lido pelo sensor.

Esse comportamento resulta em um padrão gráfico distinto: em vez de uma linha reta em torno do valor selecionado, surge um pico de irrigação que ultrapassa tal valor, evidenciando períodos de irrigação excessiva seguidos pela correção das leituras após o desligamento da válvula.

A realização desses testes permitiu observar outros fatores relevantes. Um deles é que, por ter sido montado em uma *protoboard*, o protótipo está sujeito a mau contato entre as conexões, exigindo manutenção frequente para garantir seu funcionamento adequado.

Para mitigar esse problema, é recomendável que, após a fase de prototipagem, o circuito seja transferido para uma PCB (*Printed Circuit Board*). Isso proporcionará maior confiabilidade ao sistema, minimizando o risco de desconexões indesejadas e aumentando sua robustez e durabilidade.

Para otimizar a instalação, seria necessário utilizar fios mais longos para o painel solar, permitindo sua fixação em um local com melhor incidência de luz solar. Esse problema pode ser resolvido com a utilização de uma PCB equipada com bornes, possibilitando conexões elétricas mais flexíveis.

Já em relação à comunicação MQTT, esta ocorre de maneira eficiente, garantindo uma troca de mensagens dinâmica e objetiva. O *broker* utilizado, apesar de ser público, apresenta um tempo médio de publicação inferior a 1 segundo e uma conexão relativamente rápida, estabelecendo-se em menos de 3 segundos.

A troca de mensagens entre o aplicativo e o sistema físico ocorre de forma satisfatória, sem gargalos ou interpretações errôneas por parte dos dispositivos. Essas interações podem ser observadas na Figura 26.

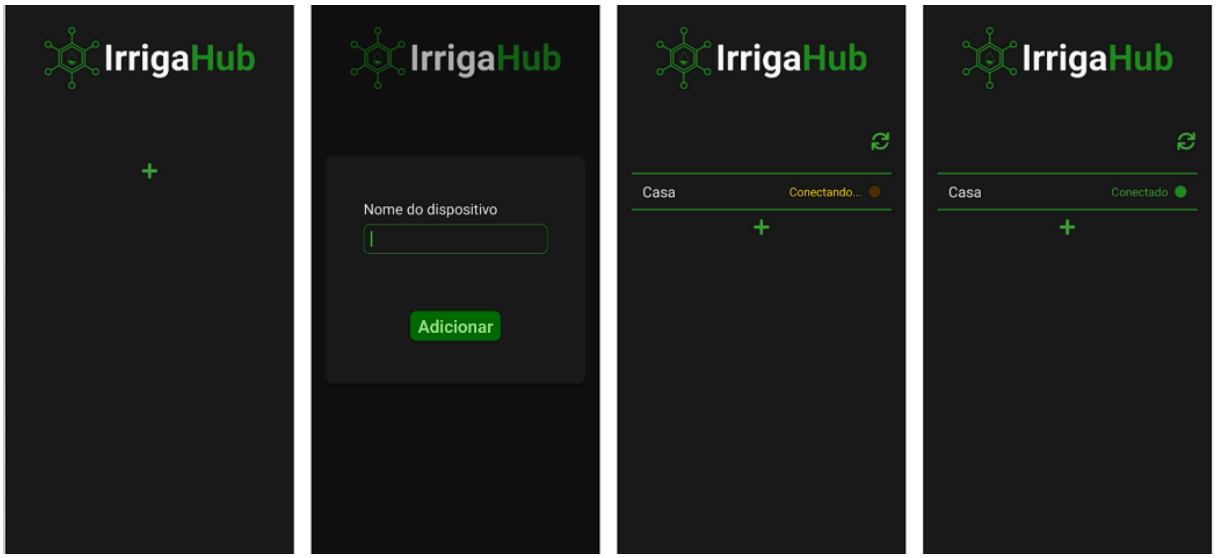
Figura 26 - Visualização mensagens *broker* MQTT


Fonte: Acervo pessoal (2025).

Além disso, o aplicativo IrrigaHub foi desenvolvido para dispositivos Android, apresentando uma interface intuitiva e um *design* agradável. Inicialmente, a compilação foi realizada apenas para Android, uma vez que a publicação para iOS (*iPhone Operating System*) requer um ambiente macOS (*Macintosh Operating System*), disponível exclusivamente em computadores da Apple, o que não foi viável devido a restrições financeiras.

Ao ser iniciado, o aplicativo exibe uma tela inicial aguardando que o usuário adicione algum dispositivo. Esse processo ocorre por meio de uma janela de cadastro, e os dispositivos adicionados são armazenados na memória cache do celular, permitindo conexões mais rápidas em acessos futuros. Essa funcionalidade pode ser observada na Figura 27.

Figura 27 - Tela inicial do aplicativo IrrigaHub demonstrando o cadastro de um novo dispositivo e sua conexão, da esquerda para a direita

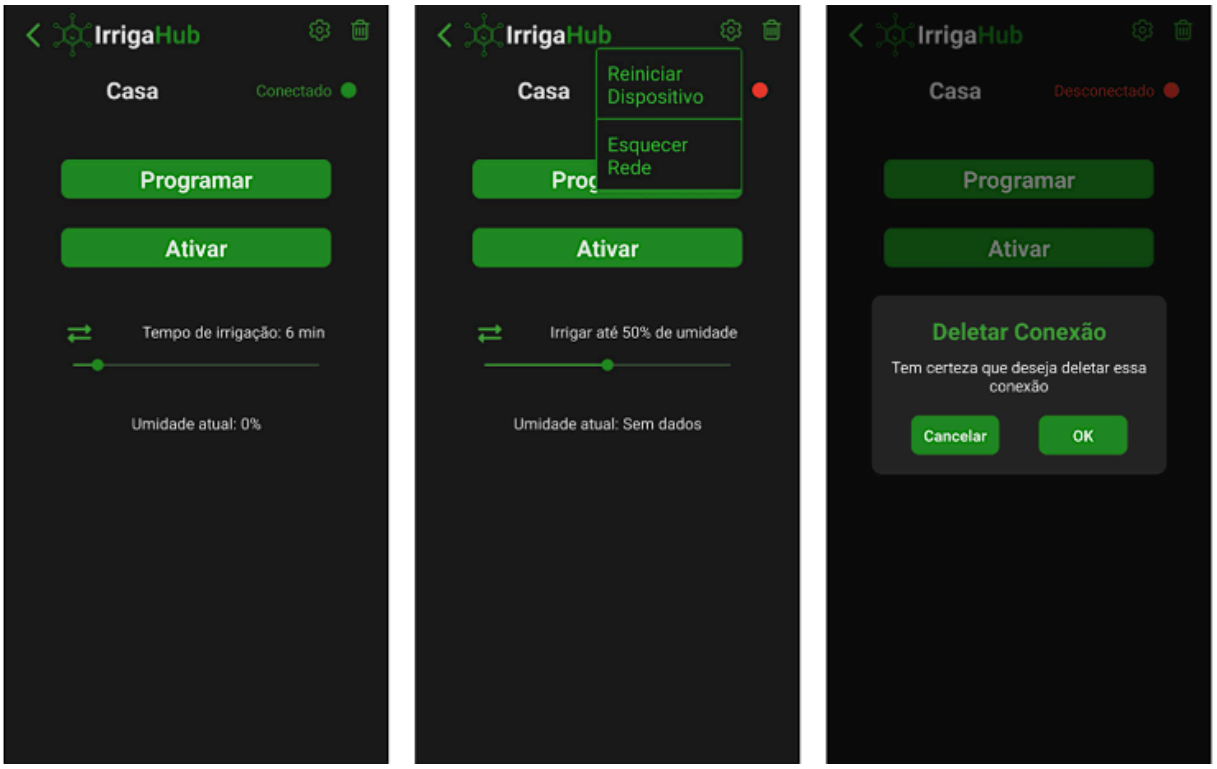



Fonte: Autoria própria (2025).

Nota-se então que o sistema permite a conexão de múltiplos dispositivos, cada um identificado por um nome distinto, viabilizando o monitoramento de diversas unidades de controle. Além disso, há um botão de atualização, possibilitando a reconexão caso o usuário identifique alguma falha na comunicação.

Ao selecionar um dispositivo, o usuário é direcionado para a tela de gerenciamento (Figura 28), que exibe um cabeçalho com três funcionalidades principais: retorno à tela inicial, opções avançadas e exclusão do dispositivo.

Figura 28 - Tela de gerenciamento do aplicativo IrrigaHub, demonstrando funcionalidade do cabeçalho



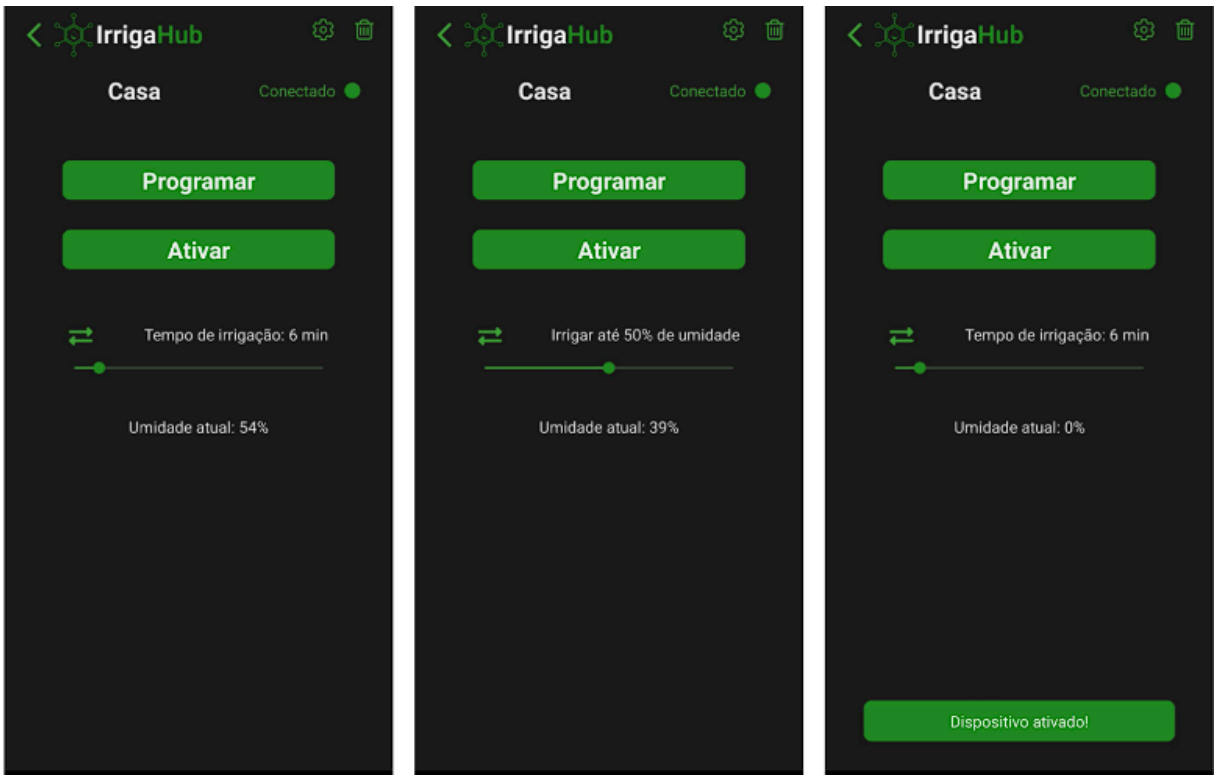

Fonte: Autoria própria (2025).

Ao pressionar na seta para a esquerda presente no cabeçalho, o aplicativo retorna à tela anterior. Já quando se pressiona na engrenagem, é aberto um menu com duas opções, sejam elas reiniciar o dispositivo ou deletar a rede Wi-Fi em que o dispositivo esteja conectado, abrindo novamente uma rede própria do dispositivo físico para conexão. Por fim, quando pressionada a lixeira, é aberta uma requisição de confirmação de ação, e então apagado da memória do celular o dispositivo atual, retornando à tela inicial.

Além disso, há também o gerenciamento de estados de conexão, possibilitando a visualização caso o dispositivo esteja enviando mensagens ao aplicativo ou não. Caso não haja conexão, não é possível enviar dados de ativação ou programação para o ESP8266.

Também estão presentes dois botões, sendo que um deles leva à tela de programação, enquanto o outro ativa naquele momento, de acordo com os parâmetros configurados pela barra deslizante, podendo ter seu parâmetro modificado, isso pode ser consultado pela Figura 29.

Figura 29 - Opções de irrigação instantânea e mensagem quando ativado

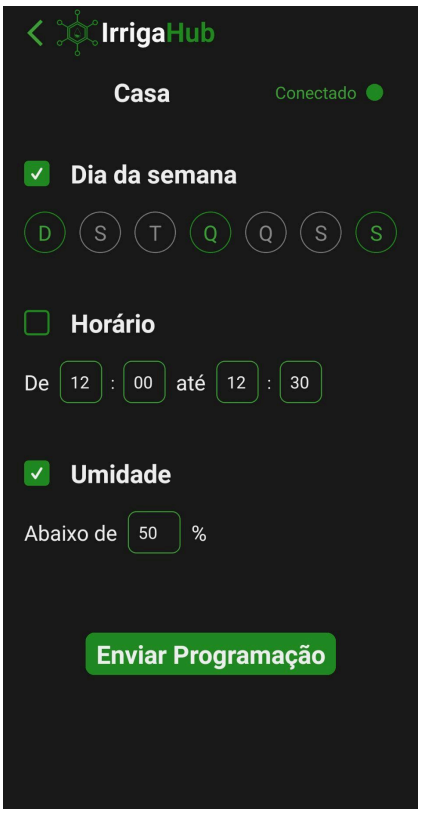



Fonte: Autoria própria (2025).

Dessa forma, de acordo com a Figura 29, a primeira imagem representa a escolha da irrigação instantânea por um período de tempo, a segunda irriga até uma porcentagem específica e, por fim, a última quando o botão de ativação for pressionado e a mensagem for enviada com sucesso. Nota-se que para realizar a troca de parâmetros basta pressionar o botão de setas duplas indicando a troca.

Por fim, a última funcionalidade do aplicativo é a tela de programação recorrente, conforme ilustrado na Figura 30.

Figura 30 - Tela de programação do aplicativo IrrigaHub




Fonte: Autoria própria (2025).

Cada opção presente nessa tela possui uma caixa de seleção que, quando selecionada, indica que aquela configuração está ativada.

Na primeira opção está a seleção do dia da semana, representados pelas iniciais de domingo a sábado, ficando destacados em verde quando selecionados. Já em relação ao horário, é um campo numérico que o usuário pode digitar o horário inicial e final, seguindo o sistema de 24 horas. E por fim, o campo de umidade também é um campo numérico de zero a cem que o usuário pode inserir o valor ideal de umidade para que o sistema irrigue até alcançá-lo.

Ao final da página há o botão responsável por realizar o envio. Quando pressionado, caso haja a conexão, é enviada a programação ao *broker* MQTT e também salva a configuração no celular do usuário.

Foi contabilizado o tempo médio de carregamento de cada tela, iniciadas cinco vezes cada e considerando um celular Xiaomi Redmi Note 8 para os testes. Para a tela inicial, houve um tempo médio de 23,84 ms, enquanto a de gerenciamento 32,93 ms e por fim, a de programação 23,98 ms. Esses tempos indicam que o aplicativo é leve, tornando esses tempos de carregamento praticamente imperceptíveis ao usuário, resultando em um bom desempenho.

Pensando na escalabilidade do aplicativo, pode-se realizar futuramente outros testes de *benchmark*, como por exemplo o consumo de CPU e RAM do dispositivo móvel, consumo de bateria e também latência média na recepção de dados. Tais testes não foram realizados no presente protótipo.

Os resultados obtidos demonstram que o sistema otimiza o uso da água, reduzindo desperdícios e proporcionando maior autonomia na gestão da irrigação. A integração entre *hardware*, MQTT e aplicativo móvel funcionou conforme esperado, garantindo comunicação eficiente, acionamento remoto confiável e interface intuitiva para o usuário.

# 6 CONCLUSÃO

O sistema desenvolvido, composto pelo protótipo físico e pelo aplicativo, atingiu os objetivos propostos, permitindo a operação por um período sem depender da rede elétrica. A utilização de *Energy Harvesting* por meio de placas solares contribui para a redução da necessidade de manutenção, enquanto o monitoramento e acionamento remoto da irrigação, aliado à possibilidade de automação via programação no aplicativo, proporciona maior autonomia ao usuário.

No entanto, a completa independência da rede elétrica não foi alcançada devido à limitação na geração de potência contínua pelas placas solares. A potência máxima de 3 W ocorre apenas sob incidência solar direta, o que impacta a disponibilidade energética ao longo do dia. Isso pode ser corrigido com a inserção de mais placas solares no sistema para aumentar a capacidade de geração, ou com a implementação de um sistema de rastreamento solar, garantindo que o painel esteja sempre orientado para a melhor captação de luz. Outra sugestão de melhoria seria realizar um estudo de incidência solar no local de instalação, permitindo o reposicionamento estratégico do painel por meio do uso de cabos mais longos.

Outro ponto de aprimoramento refere-se à estruturação do *hardware*. Para aperfeiçoamentos futuros seria ideal o desenvolvimento de uma placa de circuito impresso para comportar todos os componentes eletrônicos. Essa placa teria bornes para os cabos do sensor, da placa solar e da válvula solenóide, permitindo que sejam instalados a maiores distâncias do circuito, diminuindo também o tamanho do mesmo e consequentemente, alguns mau contatos presentes no circuito.

Além disso, a proteção física dos componentes deve ser considerada. Atualmente, a bateria fica exposta à luz solar, o que reduz sua vida útil e pode comprometer o funcionamento do sistema. Assim, uma carcaça personalizada pode ser projetada em impressão 3D para comportar o circuito e proteger a bateria e os componentes da luz solar e do sobreaquecimento. Dependendo da potência solar disponível, seria possível incluir exaustores e um fluxo de ar projetado dentro dessa carcaça para garantir o refrigeramento dos componentes internos.

Outro ponto crítico identificado foi a alimentação da válvula solenóide, que possui um consumo significativamente maior em comparação ao restante do circuito. Quando ativada, há um desbalanceamento na distribuição de energia, afetando a leitura do sensor de umidade. Dessa forma, para suprimir esse problema pode-se utilizar uma bateria independente para acionar a válvula solenóide, sendo conectada em outro carregador solar em paralelo com o

sistema do ESP. Isso evitaria os erros de leitura encontrados quando a válvula solenóide está ativada.

Já em relação ao aplicativo, os principais pontos de melhoria envolvem ativação e programação da irrigação. Atualmente não há nenhum *feedback* indicando que o sistema está irrigando, dessa forma pode ser desenvolvida uma sofisticação na comunicação de acionamento da válvula, permitindo que o aplicativo observe o estado da mesma em tempo real, decidindo não apenas quando ela deve ser ativada, mas também desativada e exibindo esse estado ao usuário.

Além disso, os parâmetros de programação atualmente são armazenados apenas no celular do usuário. Isso gera inconsistências quando múltiplos usuários acessam o mesmo dispositivo, pois não há uma forma de verificar as configurações enviadas anteriormente. Assim, a programação e parâmetros poderiam ser salvos na memória EEPROM do ESP8266, permitindo que sejam consultados quando a página for acessada e mostrados ao usuário.

Por fim, para garantir a escalabilidade do projeto, seria essencial a implementação de um servidor *backend*, banco de dados centralizado e *broker* MQTT privado. Isso possibilitaria que houvesse o cadastro de usuários e autenticação via *login*/senha. Dessa forma, seria possível vender esse projeto para diversos clientes diferentes, mantendo uma base de dados unificada e um *broker* específico para o projeto, possibilitando a proteção contra acessos indevidos, garantindo que cada usuário controle apenas seus próprios dispositivos.

Com essas melhorias, o sistema poderia ser comercializado em larga escala, atendendo desde residências até áreas agrícolas, proporcionando um controle eficiente e automatizado da irrigação.

Em suma, o projeto atendeu aos objetivos propostos, proporcionando um sistema de irrigação automatizada e monitoramento remoto, com a vantagem adicional da coleta de energia solar para reduzir a dependência da rede elétrica. Apesar de algumas limitações técnicas, o sistema demonstrou viabilidade prática e potencial para expansão.

Com as otimizações sugeridas, o IrrigaHub pode se tornar uma solução robusta para controle de irrigação inteligente, contribuindo para a eficiência no uso da água e a sustentabilidade ambiental, reduzindo desperdícios e permitindo um gerenciamento mais preciso da irrigação, tanto em aplicações domésticas quanto agrícolas

# REFERÊNCIAS

ARDUINO. **Software**. Disponível em: https://www.arduino.cc/en/software. Acesso em: 24 jan. 2025.

AWS. **O que é MQTT?** Disponível em: https://aws.amazon.com/pt/what-is/mqtt/. Acesso em: 09 fev. 2025.

BEVACQUA, Nicolás. **local-storage: A simplified localStorage API that just works**. Disponível em: https://www.npmjs.com/package/local-storage. Acesso em: 24 fev. 2025.

BHUVANESWARI, R. *et al*. **A review on energy harvesting techniques for sustainable power generation**. Energy Procedia, v. 14, p. 1519-1524, 2009.

CHEETO, M.; QUEUDET, A. Harnessing ambient energy for embedded systems. In: CHETTO, M.; QUEUDET, A. (Ed.). **Energy Autonomy of Real-Time Systems**. Elsevier, 2016, p. 57–83. [Online]. Disponível em: https://www.sciencedirect.com/science/article/pii/B9781785481253500038. Acesso em: 28 fev. 2024.

COMPANHIA AMBIENTAL DO ESTADO DE SÃO PAULO. **O problema da escassez de água no mundo**. Disponível em: https://cetesb.sp.gov.br/aguas-interiores/tpos-de-agua/o-problema-da-escasez-de-agua-no-mundo/2023. Acesso em: 13 jan. 2025.

CURTO CIRCUITO. **Sensores de umidade de solo: tipos e diferenças**. Disponível em: https://curtocircuito.com.br/blog/Categoria%20Arduino/sensores-de-umidade-de-solo-tipos-e-diferencas. Acesso em: 2 fev. 2025.

EQUIPE RAKS. **Excesso de irrigação: causas, efeitos na lavoura e como prevenir | Raks**. 29 nov. 2021. Disponível em: https://raks.com.br/excesso-de-irrigacao/. Acesso em: 28 fev. 2024.

EXPO. **Homepage**. Disponível em: https://expo.dev/. Acesso em: 2 fev. 2025.

FROST, Brad. **Atomic Design – Chapter 2.** 2016. Disponível em: https://atomicdesign.bradfrost.com/chapter-2/. Acesso em: 2 fev. 2025.

GUSE, Rosana. **Conheça o Módulo WiFi ESP8266 D1 Mini - MakerHero**. 7 fev. 2022. Disponível em: https://www.makerhero.com/blog/conheca-o-modulo-wifi-esp8266-d1-mini/. Acesso em: 2 mar. 2024.

GOKHALE, Pradyumna; BHAT, Omkar; BHAT, Sagar. **Introduction to IOT**. International Advanced Research Journal in Science, Engineering and Technology, v. 5, n. 1, p. 41-44, jan. 2018. DOI: 10.17148/IARJSET.2018.517. Disponível em: ResearchGate. Acesso em: 23 fev. 2025.

HARB, A. **Energy harvesting: State-of-the-art**. Renewable Energy, v. 36, n. 10, p. 2641-2654, 2011. Disponível em:

https://www.sciencedirect.com/science/article/pii/S0960148110003160. Acesso em: 23 fev. 2025.

HI TECNOLOGIA. **O que é o protocolo MQTT?** 24 fev. 2021. Disponível em: https://www.hitecnologia.com.br/o-que-e-o-protocolo-mqtt/. Acesso em: 09 fev. 2025.

HIVE MQ. **Homepage**. Disponível em: https://www.hivemq.com/. Acesso em: 25 jan. 2025.

ISLAM, Mohammad Minhajul; REZA, Rezawl Karim; HOSSAIN, Md. Shahadat; NATH, Anik. **IoT based automated solar irrigation system using MQTT protocol in Charandeep Chakaria**. In: INTERNATIONAL CONFERENCE ON ADVANCES IN SCIENCE, ENGINEERING AND ROBOTICS TECHNOLOGY (ICASERT), 1., 2019, Bangladesh. IEEE, 2019. Disponível em: IEEE Xplore. Acesso em: 5 fev. 2025.

MAKERHERO. **Conheça o módulo WiFi ESP8266 D1 Mini**. Disponível em: https://www.makerhero.com/blog/conheca-o-modulo-wifi-esp8266-d1-mini/. Acesso em: 2 fev. 2025.

MANEJE BEM. **Pare de enfrentar problemas com irrigação e economize água em seu cultivo! - Maneje Bem - Fitocon**. 2018. Disponível em: https://www.manejebem.com.br/publicacao/novidades/pare-de-enfrentar-problemas-com-irrigacao-e-economize-agua-em-seu-cultivo. Acesso em: 28 fev. 2024.

MICHA, Daniel. **Células e módulos fotovoltaicos**. CEFET-RJ/PUC-Rio, 2017.

NODE.JS. **Homepage**. Disponível em: https://nodejs.org/pt. Acesso em: 2 fev. 2025.

PERNAPATI, Kiranmai. **IoT based low cost smart irrigation system**. In: INTERNATIONAL CONFERENCE ON INVENTIVE COMMUNICATION AND COMPUTATIONAL TECHNOLOGIES (ICICCT), 2., 2018, Índia. IEEE, 2018. Disponível em: IEEE Xplore. Acesso em: 5 fev. 2025.

PORTAL DO GOVERNO. **Irrigar à noite é mais econômico | Governo do Estado de São Paulo**. 20 set. 2006. Disponível em: https://www.saopaulo.sp.gov.br/spnoticias/na-imprensa/irrigar-a-noite-e-mais-economico/. Acesso em: 28 fev. 2024.

RASHID, Muhammad H. **Eletrônica de potência: circuitos, dispositivos e aplicações**. 4. ed. São Paulo: Pearson, 2014.

REDAÇÃO AGRISHOW. **Você conhece as principais causas do desperdício na irrigação?** 11 jan. 2019. Disponível em: https://digital.agrishow.com.br/irrigacao/voce-conhece-principais-causas-do-desperdicio-na-irrigacao. Acesso em: 28 fev. 2024.

ROBOCORE. **Programando o ESP8266 pela Arduino IDE**. Disponível em: https://www.robocore.net/tutoriais/programando-o-esp8266-pela-arduino-ide. Acesso em: 25 jan. 2025.

ROBOCORE. **Instalando driver do NodeMCU**. Disponível em:

https://www.robocore.net/tutoriais/instalando-driver-do-nodemcu. Acesso em: 25 jan. 2025.

SANTANA, E. S.; ARENAS, L. A. O.; LIBERADO, E. V. **Energy harvesting integration with air quality monitoring systems**. Sorocaba: IEEE, 2024.

SRIVASTAVA, P.; BAJAJ, M.; RANA, A. S. **Overview of ESP8266 Wi-Fi module based Smart Irrigation System using IOT**. In: FOURTH INTERNATIONAL CONFERENCE ON ADVANCES IN ELECTRICAL, ELECTRONICS, INFORMATION, COMMUNICATION AND BIO-INFORMATICS (AEEICB), Chennai, India, 2018. Proceedings [...]. IEEE, 2018. p. 1-5. DOI: 10.1109/AEEICB.2018.8480949.

SILVA, Joana. **Tecnologia da Embrapa é usada para desenvolver sistema automático de irrigação**. 10 abr. 2018. Disponível em: https://www.embrapa.br/busca-de-noticias/-/noticia/33188097/tecnologia-da-embrapa-e-usada-para-desenvolver-sistema-automatico-de-irrigacao. Acesso em: 28 fev. 2024.

UNIUV. **Enaproc - Artigo 615**. Disponível em: https://www.periodicos.uniuv.edu.br/enaproc/article/view/615. Acesso em: 20 jan. 2025.

UNIVERSIDADE FEDERAL DO CEARÁ. **A química por trás dos painéis solares e o funcionamento de um sistema fotovoltaico**. Disponível em: http://www.petquimica.ufc.br/a-quimica-por-tras-dos-paineis-solares-e-o-funcionamento-de-um-sistema-fotovoltaico/. Acesso em: 13 jan. 2025.

VIANA, Carol. **Como utilizar o sensor de umidade do solo com o arduino – blog da robótica**. 6 ago. 2022. Disponível em: https://www.blogdarobotica.com/2022/10/06/como-utilizar-o-sensor-de-umidade-do-solo-com-o-arduino/. Acesso em: 2 mar. 2024.

YASSEIN, Muneer Bani; SHATNAWI, Mohammed Q.; ALJWARNEH, Shadi; AL-HATMI, Razan. **Internet of Things: Survey and open issues of MQTT Protocol**. In: INTERNATIONAL CONFERENCE ON ENGINEERING & MIS (ICEMIS), Monastir, Tunisia, 2017. IEEE, 2017. DOI: 10.1109/ICEMIS.2017.8273112.

# **APÊNDICE A -** Github do projeto

Código microcontrolador - https://github.com/SantiagoUnesp/IrrigaHubESP

Código aplicativo - https://github.com/SantiagoUnesp/IrrigaHubAPP