

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS
GERAIS – *CAMPUS* BAMBUÍ
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Beatriz Rodrigues de Oliveira Paiva

**DESENVOLVIMENTO DE UM SISTEMA AUTOMÁTICO
DE IRRIGAÇÃO PARA UM SETOR SELETIVO DO
VIVEIRO DO IFMG-*CAMPUS* BAMBUÍ**

BEATRIZ RODRIGUES DE OLIVEIRA PAIVA

**DESENVOLVIMENTO DE UM SISTEMA AUTOMÁTICO
DE IRRIGAÇÃO PARA UM SETOR SELETIVO DO
VIVEIRO DO IFMG-*CAMPUS* BAMBUÍ**

Monografia apresentada ao Curso de Engenharia de Computação como parte dos requisitos para a obtenção do Grau de Engenheiro de Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Bambuí

Bambuí - MG

2024

Dedico este trabalho aos meus pais, Flávia Rodrigues de Oliveira e Silva Paiva e Luiz Carlos de Paiva, e aos meus avós, Maria Aparecida Rodrigues de Oliveira, Welington Custódio de Oliveira e Silva, Arlete Sabina de Paiva e Ademalde Geraldo Paiva.

AGRADECIMENTOS

Gostaria de expressar minha gratidão, em primeiro lugar, à minha mãe, que sempre me apoiou e sem a qual nada disso seria possível. Agradeço também ao meu pai, que com suas lições contribuiu para o desenvolvimento deste trabalho. Aos meus irmãos, por me inspirarem e tornarem a caminhada mais leve e cheia de significado. À minha família, pelo amor, paciência e compreensão em todos os momentos. Aos meus amigos, que compartilharam comigo essa jornada e nunca deixaram de me apoiar. Por fim, agradeço ao meu orientador e aos meus professores, que contribuíram para o meu crescimento acadêmico e, acima de tudo, pessoal. Este trabalho é uma pequena forma de retribuir tudo o que aprendi durante esses anos e agradecer à faculdade pelos incríveis momentos vividos.

“A pesquisa é formalmente reconhecer o que todos sabemos informalmente.”
(Margaret Hamilton)

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema automatizado de irrigação para um setor do viveiro do Instituto Federal de Minas Gerais - *Campus Bambuí*, com o objetivo de otimizar o controle da irrigação e o monitoramento ambiental. A solução proposta utiliza o microcontrolador ESP32 para gerenciar o sistema e realizar a comunicação via rede sem fio. A configuração do sistema é feita por meio de uma interface *web*, acessada através de um endereço de protocolo de internet local e utilizando o protocolo de transferência de hipertexto. A interface permite ao usuário ajustar parâmetros como horário e duração da irrigação, além de monitorar, em tempo real, a temperatura ambiente e os níveis de umidade do solo e do ar. O sistema realiza o monitoramento contínuo, com dados atualizados a cada segundo pelos sensores. A irrigação é acionada automaticamente conforme horários predefinidos, e o módulo de relógio em tempo real DS1307 assegura a precisão dos tempos. O protótipo foi implementado no viveiro do Instituto Federal de Minas Gerais, com a integração de sensores e dispositivos para o controle da irrigação, contribuindo para a eficiência no cultivo das mudas.

Palavras-chave: Automação. Sistema de Irrigação. ESP32. Sensores. Monitoramento via *internet*.

ABSTRACT

This study presents the development of an automated irrigation system for a sector of the nursery at the Federal Institute of Minas Gerais - Campus Bambuí, aiming to optimize irrigation control and environmental monitoring. The proposed solution uses the ESP32 microcontroller to manage the system and communicate via a wireless network. System configuration is performed through a web interface, accessible via a local internet protocol address and using the hypertext transfer protocol. The interface allows users to adjust parameters such as irrigation schedule and duration, as well as monitor, in real-time, ambient temperature and soil and air humidity levels. The system continuously monitors with sensor data updated every second. Irrigation is automatically activated according to predefined schedules, and the DS1307 real-time clock module ensures precise timing. The prototype was implemented in the nursery at the Federal Institute of Minas Gerais, integrating sensors and devices for irrigation control, contributing to increased efficiency in seedling cultivation.

Keywords: Automation. Irrigation System. ESP32. Sensors. Internet-based Monitoring.

LISTA DE FIGURAS

Figura 1 – Setor três da estufa.	20
Figura 2 – Arquitetura de um microcontrolador.	24
Figura 3 – Pinagem dos componentes internos do ESP32.	25
Figura 4 – Interface do Arduino IDE.	28
Figura 5 – Sensor de Umidade YL-69.	30
Figura 6 – Sensor de Temperatura e Umidade DHT11.	31
Figura 7 – Relé modular utilizado no projeto.	32
Figura 8 – Módulo relé utilizado no projeto.	33
Figura 9 – Módulo RTC utilizado no projeto.	33
Figura 10 – Interface do <i>software</i> EasyEDA.	35
Figura 11 – Fluxograma da estrutura do viveiro IFMG.	40
Figura 12 – Diagrama de caso de uso do sistema desenvolvido.	40
Figura 13 – Antes e depois da válvula de irrigação instalada no viveiro.	45
Figura 14 – Componentes do sistema de irrigação.	45
Figura 15 – Equipamentos antes e depois da troca dos fios.	46
Figura 16 – Esquema elétrico da rede com o X-Core.	47
Figura 17 – Esquema elétrico da rede com a substituição do X-Core pelo protótipo.	48
Figura 18 – Painéis elétricos da sala de controle em que o X-Core estava instalado.	49
Figura 19 – Conexão do sensor no x-core.	50
Figura 20 – Uso do Wokwi para a primeira versão do protótipo e código.	52
Figura 21 – ESP32 conectado aos equipamentos no <i>protoboard</i>	53
Figura 22 – Circuito esquemático do projeto elaborado no Easyeda.	54
Figura 23 – Circuito impresso desenvolvido no Easyeda para impressão na placa.	55
Figura 24 – Transferência do desenho no papel para a placa de circuito impresso.	56
Figura 25 – Placa de fenolite após a transferência do desenho e a limpeza do excesso de papel.	57
Figura 26 – Corrosão da placa no percloroeto de ferro.	57
Figura 27 – Placa de fenolite após a soldagem dos componentes.	58
Figura 28 – Protótipo finalizado.	59
Figura 29 – Parte da página <i>web</i> com o monitoramento dos sensores e dos horários da rega.	72
Figura 30 – Parte da página HTML gerada pela função <i>handle root</i>	74
Figura 31 – Interface <i>Web</i> do protótipo.	77
Figura 32 – Sensor de umidade do solo após a instalação.	78
Figura 33 – Protótipo instalado no painel elétrico.	79
Figura 34 – Utilização de um <i>notebook</i> para acessar a interface <i>web</i> e realizar os testes necessários.	79

LISTA DE TABELAS

Tabela 1 – Descrição da Bomba Centrífuga Monoestágio Schneider BC-91 S. . . .	21
Tabela 2 – Descrição da Válvula Elétrica 100-DVF 1"Rain Bird.	21
Tabela 3 – Especificações do Controlador X-Core.	22
Tabela 4 – Especificações e Pinagem do sensor YL-69.	30
Tabela 5 – Especificações do Sensor de Temperatura e Umidade DHT11.	31
Tabela 6 – Especificações do Módulo RTC DS1307 + EEPROM 24C32 I2C. . . .	34
Tabela 7 – Orçamento dos Materiais e Equipamentos.	43

LISTA DE CÓDIGOS

Código 1 – Bibliotecas e definições dos pinos - primeiro código.	60
Código 2 – Função <i>setup</i> - primeiro código.	60
Código 3 – Função <i>loop</i> - primeiro código.	61
Código 4 – Definição das bibliotecas e dos pinos - segundo código.	62
Código 5 – Definição das variáveis - Segundo Código.	62
Código 6 – Definição da função para ligar a rega automática - segundo código. . .	63
Código 7 – Definição da função para ligar a rega manual - segundo código. . . .	63
Código 8 – Definição da função para desligar a rega manual - segundo código. . .	64
Código 9 – Função para calibrar o sensor de umidade do solo - segundo código. . .	64
Código 10 – Parte da função (<i>updateSensorsValue</i>) que recebe as leituras dos senso- res para atualizar os valores dos sensores - segundo código.	65
Código 11 – Parte da função (<i>updateSensorsValue</i>) que converte os valores - segundo código.	65
Código 12 – Configuração dos pinos e inicialização dos dispositivos na função <i>setup</i> - segundo código.	66
Código 13 – função <i>loop</i> - segundo código.	66
Código 14 – Definição da conexão com a internet - terceiro código.	68
Código 15 – função para controlar a rega de acordo com as horas - terceiro código.	68
Código 16 – Liga e Desliga a rega manualmente - terceiro código.	69
Código 17 – Função que gera a página em HTML para acesso remoto - terceiro código.	70
Código 18 – Seção dos sensores e horários, da função <i>handle root</i> - terceiro código. .	71
Código 19 – Seção de controle da rega, da função <i>handle root</i> - terceiro código. . .	71
Código 20 – Seção de configurações do sistema - terceiro código.	73
Código 21 – Seção de controle manual - terceiro código.	73
Código 22 – Função que processa as configurações de horários - terceiro código. . .	75
Código 23 – Função responsável por processar as configurações manuais da rega - terceiro código.	75
Código 24 – Definição da conexão com a internet dentro da função <i>Setup</i> - terceiro código.	76
Código 25 – Função <i>loop</i> - terceiro código.	76

LISTA DE ABREVIATURAS E SIGLAS

ADC - *Analog-to-Digital Converter*

CLK - *Clock*

CPU - *Central Processing Unit*

CSS - *Cascading Style Sheets*

DAC - *Digital-to-Analog Converter*

DSR - *Design Science Research*

EDA - *Electronic Design Automation*

EN - *Enable*

EEPROM - *Electrically Erasable Programmable Read-Only Memory*

GND - *Ground*

GPIO - *General Purpose Input/Output*

HTML - *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

I2C - *Inter-Integrated Circuit*

IDE - *Integrated Development Environment*

IFMG - Instituto Federal de Minas Gerais

IoT - *Internet of things*

IP - *Internet Protocol*

LED - *Light Emitting Diode*

PCB - *Printed Circuit Board*

RAM - *Random Access Memory*

ROM - *Read-Only Memory*

RTC - *Real-Time Clock*

SPI - *Serial Peripheral Interface*

SPICE - Programa de Simulação com Ênfase em Circuito Integrado

SSID - *Service Set Identifier*

V_{in} - *Voltage Input*

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Caracterização do problema	15
1.2	Objetivos	16
1.3	Justificativa	17
1.4	Organização do trabalho	17
2	REFERENCIAL TEÓRICO	19
2.1	Ambiente de aplicação do trabalho	19
2.1.1	<i>Viveiro</i>	19
2.1.2	<i>Equipamentos</i>	20
2.1.2.1	Bomba	20
2.1.2.2	Válvula	21
2.1.2.3	Transformador	21
2.1.2.4	X-Core	22
2.2	Sistemas Embarcados	22
2.3	Microcontrolador	23
2.3.1	<i>ESP32</i>	24
2.3.2	<i>Linguagem de programação</i>	27
2.3.3	<i>Arduino IDE</i>	27
2.4	Sensores	28
2.4.1	<i>Sensor de Umidade YL-69</i>	29
2.4.2	<i>Sensor de Temperatura e Umidade do Solo DHT11</i>	30
2.5	Relé	31
2.6	Módulo RTC	33
2.7	EasyEda	34
2.8	Estado da arte	35
3	METODOLOGIA	38
3.1	Classificação do trabalho	38
3.2	Solução Proposta	39
3.2.1	<i>Materiais, Equipamentos e Tecnologias</i>	41
3.2.2	<i>Orçamentos</i>	43
4	DESENVOLVIMENTO	44
4.1	Identificação e reparo dos equipamentos existentes	44
4.2	Identificação e reparo da rede elétrica	46
4.3	Inspeção no X-core	48

5	RESULTADOS	51
5.1	Protótipo	51
5.1.1	<i>Concepção inicial do projeto</i>	51
5.1.2	<i>Montagem no protoboard</i>	52
5.1.3	<i>Arduino IDE</i>	53
5.1.4	<i>Software Easyeda para projeto da placa</i>	54
5.1.5	<i>Montagem da placa</i>	55
5.2	Código	59
5.2.1	<i>Controle do relé</i>	60
5.2.2	<i>Monitoramento dos sensores</i>	61
5.2.3	<i>Controle pela web</i>	67
5.3	Instalação	77
6	CONCLUSÃO	80
6.1	Trabalhos Futuros	80
	REFERÊNCIAS	81
	APÊNDICE	85

1 INTRODUÇÃO

A agricultura é uma atividade que visa a produção de alimentos, fibras e outros produtos para sustentar a população mundial. Com o avanço das tecnologias, métodos modernos, como a agricultura de precisão, têm ajudado a otimizar o uso dos recursos, aumentando a produtividade e reduzindo o impacto ambiental (BASSO; ANTLE, 2020).

A irrigação é a técnica de aplicação de água nas plantas de forma controlada para promover seu crescimento, sendo usado em áreas com disponibilidade limitada de água. Métodos modernos de irrigação, como a irrigação com aspersores e por gotejamento, buscam fornecer a quantidade de água ideal para cada planta, reduzindo o desperdício e melhorando a eficiência hídrica (SAVAGE, 2023).

Estufas são estruturas fechadas que permitem o cultivo de plantas em ambientes controlados, protegendo-as de condições climáticas adversas (DEOGIRIKAR, 2021). Em uma estufa, fatores como temperatura, umidade e luz podem ser ajustados conforme a necessidade das culturas, promovendo um crescimento mais eficiente e independente do clima externo.

1.1 Caracterização do problema

Atualmente, o viveiro de produção de mudas do Instituto Federal de Minas Gerais (IFMG) - *Campus* Bambuí é dividido em seis setores distintos. Cinco deles são dedicados ao cultivo de plantas, enquanto o sexto abriga uma lagoa, localizada à direita na entrada do *campus*.

Das áreas destinadas ao cultivo de plantas, duas são abertas e três estão dentro de uma estufa coberta. Cada uma dessas áreas apresenta diferentes necessidades para o desenvolvimento adequado das mudas, especialmente em relação à irrigação.

A estufa destinada ao cultivo de plantas é subdividida em três setores, cada um voltado ao cultivo de diferentes espécies.

Durante o desenvolvimento deste trabalho, o primeiro setor da estufa abrigava plantas voltadas para paisagismo e ornamentação, como suculentas, cactos e orquídeas. Essas espécies possuem uma demanda hídrica inferior em comparação com as demais (SALACHNA; WILAS; ZAWADZIŃSKA, 2014).

No segundo setor encontram-se as matrizes responsáveis pela produção de mudas de espécies aromáticas e medicinais, destinadas tanto à comercialização quanto à doação. Nesse espaço, são cultivadas plantas como manjeriço, hortelã, bálsamo, alecrim, boldo e orégano, que apresentam alta demanda hídrica para o desenvolvimento adequado das mudas (SANTOS, 2021).

O terceiro setor da estufa é destinado a atividades de pesquisa e extensão, abrigando espécies vegetais destinadas tanto à comercialização quanto à doação. Este setor foi escolhido para o presente estudo devido à sua disponibilidade e à adequação às

necessidades da pesquisa (LIMA PEREIRA *et al.* 2021). Além disso, o local já contava com estrutura parcial e equipamentos que facilitaram a implementação do projeto.

Foi observado que a estufa contava com um sistema de irrigação automatizado, composto por aspersores para irrigar as plantas. O microcontrolador utilizado era o X-Core. Este equipamento estava danificado e não estava conectado aos demais dispositivos. Logo, o sistema automatizado estava inoperante (SILVA; CARVALHO, 2023).

Com o sistema inoperante, a irrigação vinha sendo realizada manualmente por uma servidora do IFMG, duas vezes ao dia, apenas em dias úteis. No entanto, a rega manual não é ideal, pois não cobre o final de semana nem os feriados e pode levar a desperdício de água e prejudicar a produção das plantas (GOMES; LIMA, 2023). Além disso, alguns equipamentos instalados no setor apresentavam avarias.

Diante dos problemas identificados no viveiro, este trabalho propõe o desenvolvimento de um sistema automatizado para um dos setores. Esse sistema permitirá, por meio de uma interface *web*, monitorar os sensores de umidade do ar, umidade do solo e de temperatura, além de controlar o horário de irrigação.

A adoção de um sistema de irrigação automatizado em estufas traz algumas vantagens, como o aumento da produção e a redução de desperdícios, devido ao controle preciso da quantidade de água necessária para cada planta (SILVA; LOURENÇO; NASCIMENTO, 2022). Estudos indicam que esses sistemas não apenas facilitam o manejo em grandes áreas, mas também ajuda na eficiência da produção agrícola ao longo do ano, independentemente de fatores climáticos externos ((MENDES; PEREIRA, R. T., 2023); (OLIVEIRA; SOUZA, B. L., 2021)).

1.2 Objetivos

O objetivo principal do presente trabalho é desenvolver um sistema automatizado de irrigação para um setor específico do viveiro do IFMG - *Campus* Bambuí (Setor 3 da estufa), que possibilite o monitoramento dos parâmetros ambientais (umidade do ar, umidade do solo e temperatura) e controle das regas por meio de dispositivos conectados à *internet*, de modo a aumentar a eficiência no processo de irrigação das mudas.

Já os objetivos específicos do trabalho são:

- Realizar uma revisão bibliográfica sobre o tema.
- Fazer um levantamento sobre os equipamentos utilizados, além de uma inspeção nos equipamentos existentes no Setor 3 do viveiro.
- Desenvolver um protótipo de sistema microcontrolado em bancada.
- Desenvolver um sistema *web* para monitorar os parâmetros enviados pelos sensores.
- Realizar as adequações necessárias e instalar o sistema microcontrolado desenvolvido.

1.3 Justificativa

A irrigação é um fator importante para garantir a sobrevivência e o crescimento adequado das plantas em um viveiro (FACHINELLO; HOFFMANN; NACHTIGAL *et al.* 2005). No Setor 3 do IFMG - *Campus* Bambuí, uma servidora realiza a rega manualmente. Esse método é ineficiente e pode causar regas excessivas ou insuficientes, resultando em desperdício de água, surgimento de doenças e até perda de mudas. Além disso, a ausência de um sistema automatizado de irrigação demanda um significativo dispêndio de mão de obra, pois a irrigação ocorre duas vezes todos os dias úteis do ano (MALLAREDDY *et al.* 2023).

Diante disto, este trabalho visa implementar um sistema de irrigação automatizado para melhorar a qualidade de vida das plantas no Setor 3 do viveiro do IFMG - *Campus* Bambuí. A utilização de um sistema automatizado para irrigação pode reduzir o desperdício de água, de energia elétrica e de mão de obra, contribuindo para a sustentabilidade ambiental ((SANTOS, C. A.; OLIVEIRA, 2022) e (ADEBAYO *et al.* 2023))

1.4 Organização do trabalho

Este texto está organizado da seguinte forma:

- **Capítulo 1 - Introdução:** Introduz os conceitos fundamentais de agricultura, a importância da irrigação e as especificidades do uso de estufas para cultivo controlado. Caracteriza o problema da irrigação em estufas, abordando os desafios de monitoramento e controle eficientes das variáveis ambientais. Descreve o ambiente de aplicação e as características da estufa utilizada, assim como os objetivos e a justificativa para o desenvolvimento do sistema automatizado, ressaltando os potenciais benefícios de sua implementação.
- **Capítulo 2 - Referencial Teórico:** Fornece uma base conceitual e técnica para o trabalho, abordando o ambiente de aplicação e os equipamentos do sistema. Explica o conceito de sistemas embarcados e do microcontrolador, detalha a linguagem de programação utilizada e o ambiente de desenvolvimento. Descreve os principais dispositivos empregados no protótipo e discute o software usado no desenvolvimento do circuito impresso. Apresenta uma revisão das referências teóricas fundamentais que sustentam o projeto e contextualiza o trabalho frente a outras inovações na área.
- **Capítulo 3 - Metodologia:** Descreve a metodologia adotada para o desenvolvimento e implementação do projeto, baseada nos princípios da *Design Science Research* (DSR), delineando as etapas e os objetivos da pesquisa para a criação da solução. Apresenta a solução implementada, a abordagem adotada e a estruturação

da proposta, incluindo uma visão geral dos materiais, equipamentos e tecnologias utilizados no projeto. Justifica as escolhas técnicas e metodológicas, além de incluir a análise dos custos envolvidos.

- **Capítulo 4 - Desenvolvimento:** Descreve as etapas do desenvolvimento do protótipo, desde a concepção inicial até a implementação final. Inclui a identificação e reparo dos equipamentos existentes, revisão da rede elétrica e inspeção do X-Core.
- **Capítulo 5 - Resultados:** Detalha a montagem da placa e desenvolvimento do código de controle. Relata a instalação do sistema no viveiro e a verificação de seu funcionamento.
- **Capítulo 6 - Conclusão:** Resume o desenvolvimento do sistema enfatizando seus resultados. Destaca a eficácia do protótipo em testes práticos. Inclui a importância do projeto para a agricultura de precisão e gestão sustentável. Apresenta sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este Capítulo aborda os principais temas relacionados ao estudo em questão. Inicia-se com uma análise do ambiente de aplicação, destacando o viveiro e os equipamentos essenciais ao sistema na Seção 2.1.

Na sequência, uma breve explicação sobre sistemas embarcados na Seção 2.2, em que, são explorados os conceitos fundamentais dessa área. A Seção 2.3 detalha os microcontroladores. Uma descrição dos principais dispositivos utilizados no protótipo é feita, sensores (Seção 2.4), relés (Seção 2.5) e o na Seção 2.6 o módulo relógio em tempo real - RTC (sigla do inglês *Real-Time Clock*). Uma descrição do *software* empregado no desenvolvimento da placa de circuito impresso é mostrada na Seção 2.7. Por fim, a Seção 2.8 apresenta uma revisão do estado da arte, contextualizando as contribuições e inovações relevantes na área.

2.1 Ambiente de aplicação do trabalho

Nesta Seção será feito uma descrição do viveiro, abordando suas características e organização (Subseção 2.1.1), além de uma explicação sobre os equipamentos utilizados e suas respectivas funções no contexto do projeto (Subseção 2.1.2).

2.1.1 Viveiro

Os viveiros são espaços com características específicas, dedicados à produção e ao cuidado de plantas jovens até que atinjam a idade e o tamanho adequados para serem transferidas para o local definitivo (WENDLING, I.; FERRARI; GROSSI, 2002).

As estufas são instalações projetadas para cultivar e propagar plantas em condições controladas. Cobertas por um material transparente, como vidro ou plástico especial, permitem a passagem de luz solar e retêm o calor interno. Essas estruturas protegem as plantas de ameaças e condições climáticas adversas, mantendo a temperatura adequada no ambiente (REIS *et al.* 2005).

O trabalho foi implementado no viveiro de produção de mudas do IFMG - *Campus* Bambuí, especificamente no Setor 3. Este setor integra uma estufa coberta com plástico. Nesse setor, encontram-se diversas plantas, incluindo espécies como cactos e suculentas, utilizadas para estudos e comercialização. A Figura 1 apresenta uma vista interna do setor três, ilustrando a disposição dessas espécies e o ambiente criado para seu cultivo.

Figura 1 – Setor três da estufa.



Fonte: Elaborado pela autora, 2024.

2.1.2 Equipamentos

Para iniciar o presente estudo, foi realizada uma visita à estufa com o objetivo de conhecer o local, entender suas demandas e identificar os equipamentos já existentes no setor. Embora a estufa já possuísse um sistema de irrigação instalado, os equipamentos estavam inoperantes por motivos diversos.

A descrição dos equipamentos encontrados no Setor 3 será apresentada nas subseções seguintes.

2.1.2.1 Bomba

A bomba elétrica é um dispositivo utilizado para transportar líquidos de um local para outro. Ela possui um motor elétrico que aciona o mecanismo de bombeamento, geralmente composto por um rotor e um impulsor. Quando acionado, o motor elétrico fornece a energia necessária para a rotação do rotor, que movimenta o impulsor. Esse movimento cria um vácuo que aspira a água da caixa d'água. À medida que a água é aspirada pela bomba, ela é direcionada para a câmara de bombeamento, onde ocorre um aumento de pressão. A pressão gerada pelo movimento das pás impulsiona o líquido para fora da bomba, através da tubulação de descarga (RODYCZ; VIEIRA, 2015).

A bomba instalada na estufa é o modelo BC-91 S, responsável por transportar a água necessária para a irrigação. Em boas condições de uso, o equipamento demonstrou eficiência ao distribuir água para as válvulas solenoides, garantindo o fluxo adequado para o sistema de irrigação. Na Tabela 1, são apresentadas algumas especificações da bomba, retiradas do site do fabricante¹.

¹ <https://schneider.ind.br/produtos/motobombas-de-superfície>

Tabela 1 – Descrição da Bomba Centrífuga Monoestágio Schneider BC-91 S.

Característica	Descrição
Potência	1 CV
Fases	Trifásica
Tensão	220V/380V
Rotação	3.500 rpm (2 Pólos)
Temperatura máxima permitida	70°C

Fonte: Adaptado do site do fabricante, 2024.

2.1.2.2 Válvula

A válvula é um equipamento que opera com base em um solenoide, que é um componente formado por uma bobina de fio condutor enrolada em torno de um núcleo ferromagnético. Quando a corrente elétrica passa pela bobina do solenoide, ela cria um campo magnético que atrai o núcleo ferromagnético para dentro da bobina (TREMPEL, 2015). Esse movimento do núcleo abre a válvula, permitindo que a água passe através dela e seja direcionada aos aspersores.

A válvula utilizada foi projetada para sistemas de irrigação residenciais e públicos de menor dimensão. Ela é do modelo 100-DVF 1" da marca Rain Bird. Algumas informações retiradas do site do fabricante² podem ser visto na Tabela 2.

Tabela 2 – Descrição da Válvula Elétrica 100-DVF 1" Rain Bird.

Característica	Descrição
Marca	Rain Bird
Modelo	100-DVF
Dimensões (cm)	Largura: 14, Altura: 11, Comprimento: 17
Temperatura da água	Até 110°F (43°C)
Potência do solenoide	24V

Fonte: Adaptado do site do fabricante, 2024.

2.1.2.3 Transformador

Na estufa, está instalado um transformador, responsável por ajustar a tensão e a corrente elétrica para os dispositivos, como a válvula, que operam com menor intensidade em comparação à bomba.

O transformador converte a tensão elétrica de entrada para os valores necessários ao funcionamento dos equipamentos. Assim, o equipamento reduz a tensão elétrica de 220V para 24V, fornecendo energia à válvula e a outros dispositivos que exigem essa tensão reduzida.

² <https://www.rainbird.com/pt-pt/products/valvulas-das-series-dvdf>

Segundo Alexander e Sadiku (2021), o funcionamento básico de um transformador baseia-se no princípio da indução eletromagnética. O transformador consiste em duas bobinas de fio condutor enroladas em torno de um núcleo ferromagnético. A bobina conectada à fonte de energia é chamada de primária, enquanto a bobina conectada ao circuito de saída é chamada de secundária. Quando uma corrente alternada passa pela bobina primária, ela cria um campo magnético variável. Este campo magnético induz uma corrente elétrica na bobina secundária, conforme descrito pela Lei de Faraday da indução eletromagnética.

2.1.2.4 X-Core

O X-Core é um equipamento microcontrolado projetado para gerenciar a irrigação de forma inteligente e simples. O dispositivo estava inoperante e foi necessário realizar manutenções para restaurar seu funcionamento.

De acordo com o manual do equipamento³, o X-Core permite a programação de até 3 programas de rega distintos. Cada programa pode ser configurado com até 4 horários de início por dia. Isso permite agendar até 12 inícios de rega diários. Além disso, o X-Core oferece opções, como regas intervaladas (por exemplo, a cada 2 dias), rega par/ímpar (em dias pares ou ímpares).

As especificações elétricas do X-Core estão detalhadas na Tabela 3.

Tabela 3 – Especificações do Controlador X-Core.

Característica	Descrição
Entrada do Transformador	120 VAC $\pm 10\%$ 60 Hz
Saída do Transformador	24 VAC 1.0 ampere
Bateria	Bateria de lítio de 3 V (CR2032)
Proteção Eletrônica	Proteção contra curto-circuito
Memória	Memória não volátil para armazenamento de dados de programa

Fonte: Adaptado do manual do equipamento, 2024.

Atualmente, todos os equipamentos listados estão funcionando adequadamente. O equipamento X-Core foi substituído pelo protótipo desenvolvido durante o trabalho. A seguir, serão apresentados conceitos gerais sobre sistemas embarcados.

2.2 Sistemas Embarcados

Os sistemas embarcados são dispositivos projetados para realizar uma função específica de forma contínua e eficiente. Eles diferem de dispositivos gerais, como *smartphones* e computadores pessoais, que são projetados para executar uma variedade de tarefas

³ <https://www.hunterindustries.com/sites/default/files/LIT-397-RevG-OM-X-CORE-EN-web.pdf>

(HEMACHANDRA, 2023). Em contraste, os sistemas embarcados são dedicados a uma tarefa específica e são otimizados para funcionar dentro das restrições de seu ambiente (WANG *et al.* 2021).

Esses sistemas oferecem um bom custo-benefício, que resulta de seu tamanho compacto e baixo consumo de energia. A integração de sistemas embarcados em dispositivos específicos proporciona uma operação eficiente do projeto desenvolvido (HARVIE, 2023).

O sistema de irrigação proposto no projeto é considerado um sistema embarcado. O microcontrolador é o componente central que executa as funções necessárias, garantindo a operação contínua e eficaz do sistema.

2.3 Microcontrolador

O microcontrolador é um dispositivo eletrônico integrado que inclui um processador, memória e periféricos de entrada/saída em um único *chip*. A junção destes componentes proporciona que o microcontrolador execute tarefas específicas de controle e processamento de dados em sistemas embarcados (BARRETT, 2022).

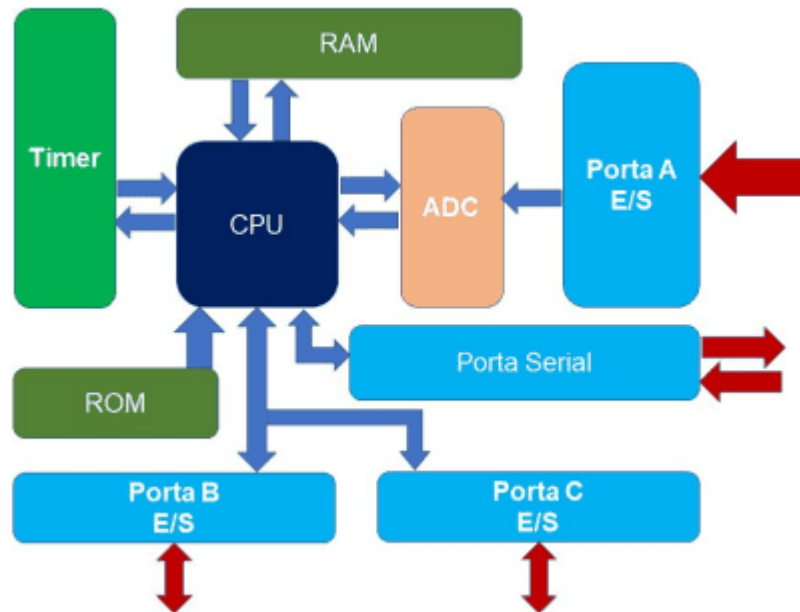
O Professor Ricardo Kerschbaumer define microcontrolador como sendo:

Microcontroladores são circuitos integrados que possuem em seu interior todos os componentes necessário ao seu funcionamento dependendo unicamente da fonte de alimentação externa. Pode-se dizer que os microcontroladores são computadores de um único *chip*. (KERSCHBAUMER, 2021)

JUNIOR (2013) afirma que a principal característica do microcontrolador é reunir, em um único *chip*, todos os periféricos necessários para o projeto e a fabricação de dispositivos eletrônicos com as mais variadas funcionalidades. Isso permite desenvolver desde simples sinalizadores ou luzes pisca-pisca até equipamentos médicos sofisticados.

Importante destacar que um microcontrolador típico possui como componentes uma unidade central de processamento (*Central Processing Unit – CPU*), uma memória de programa (*Read-Only Memory – ROM*, ou *flash*), uma memória de dados (*Random Access Memory – RAM*), além de periféricos integrados, tais como portas de entrada/saída (*General Purpose Input/Output – GPIO*), conversores analógico-digitais (*Analog-to-Digital Converter – ADC*) e temporizadores (*timers*). A Figura 2 mostra a arquitetura básica de um microcontrolador com estes componentes.

Figura 2 – Arquitetura de um microcontrolador.



Fonte: Adaptado de (SCHUBERT, 2020).

Em geral, a CPU de um microcontrolador executa as instruções do *software* armazenadas na memória de programa, que pode ser do tipo somente leitura (ROM) ou regravável (*flash*), permitindo a atualização do *firmware* do dispositivo. A memória de dados é utilizada para armazenar temporariamente informações durante a execução das instruções (REESE; BRUCE; A., 2014).

Os periféricos integrados de um microcontrolador fornecem interfaces que permitem sua comunicação com o ambiente externo. As portas de entrada/saída possibilitam a conexão do microcontrolador a dispositivos externos, como sensores e atuadores. Os conversores analógico-digitais permitem que o microcontrolador leia sinais analógicos, como temperatura e luminosidade, convertendo-os em valores digitais que podem ser processados pelo *software*. Já os *timers* são utilizados para gerar pulsos de *clock*, temporizadores e interrupções periódicas.

Os microcontroladores geralmente são programados utilizando linguagens de programação de alto nível, como C++, que foi empregada neste projeto. O microcontrolador escolhido foi o ESP32, devido às suas especificações avançadas, conectividade *Wi-Fi* e suporte a múltiplos sensores e atuadores. Na próxima Seção, serão apresentados detalhes técnicos e funcionais sobre esse dispositivo.

2.3.1 ESP32

O ESP32 pode ser definido como um microcontrolador que possui uma alta capacidade de processamento de dados. Desenvolvido pela empresa chinesa Espressif

Integrado - I2C (sigla do inglês *Inter-Integrated Circuit*), que promove a comunicação serial síncrona entre o ESP32 e outros dispositivos; temporizadores, que permitem a execução de tarefas em intervalos específicos; RTC, que fornece informações sobre data e hora, mesmo quando o microcontrolador está em modo de baixo consumo de energia; *Wi-Fi*, que permite a conexão à internet e a comunicação sem fio com outros dispositivos na rede; e *Bluetooth*, que possibilita a comunicação de curto alcance com dispositivos compatíveis.

O Quadro 1 apresenta um comparativo técnico entre três microcontroladores amplamente utilizados em sistemas de automação: ESP32, PIC e Arduino ((DIDI; EL AZAMI, 2022) e (SAMIULLAH; IRFAN; RAFIQUE, 2023)).

Quadro 1 – Comparação entre ESP32, PIC e Arduino.

Característica	ESP32	PIC (ex. PIC16F877A)	Arduino (ex. Uno)
Preço Aproximado	R\$ 43,00	R\$ 30,00	R\$ 50,00
Processador	Dual-core Tensilica LX6, 240 MHz	Single-core 8-bit, 20 MHz	ATmega328P, 8-bit, 16 MHz
Capacidade de Memória	520 KB RAM, 4 MB flash	368 Bytes RAM, 256 Bytes EE-PROM	2 KB RAM, 32 KB flash
Conectividade <i>Wi-Fi</i> e <i>Bluetooth</i> integrados	Sim	Não	Não
Vantagens	Alta capacidade de processamento; conectividade integrada (<i>Wi-Fi/Bluetooth</i>)	Baixo consumo de energia; custo baixo; simplicidade de uso em sistemas embarcados	Simplicidade de programação; amplo suporte da comunidade; ideal para iniciantes
Desvantagens	Consome mais energia que PIC e Arduino	Processamento limitado; não possui conectividade integrada; dificuldade para escrever código	Menor capacidade de processamento e memória que ESP32; sem conectividade integrada

Fonte: Elaborado pela autora, 2024.

O ESP32 destaca-se por ser um microcontrolador com processador *dual-core* de 240 MHz, além de integrar *Wi-Fi* e *Bluetooth*. Além disso, ele pode ser programado pelo Arduino IDE - Ambiente de Desenvolvimento Integrado (sigla do inglês *Integrated Development Environment*), o que facilita seu uso em projetos.

O PIC, por sua vez, possui diversas famílias de microcontroladores com capacidades variadas. Alguns modelos de PIC são adequados para sistemas embarcados mais simples, enquanto outros oferecem maior capacidade de processamento. No entanto, de

forma geral, o PIC não possui conectividade integrada e pode ser considerado mais difícil de programar em comparação com o Arduino ou o ESP32.

Já o Arduino, embora tenha limitações em termos de processamento e memória e não possua conectividade integrada, é amplamente popular pela sua facilidade de programação e pela grande comunidade de suporte. Isso o torna ideal para prototipagem rápida e projetos educacionais.

Entre suas principais vantagens estão o processador de alta capacidade, a conectividade *Wi-Fi* e *Bluetooth* integradas, o que elimina a necessidade de módulos adicionais. Além disso, o ESP32 oferece uma excelente capacidade de armazenamento e execução de programas. Embora o ESP32 consuma um pouco mais de energia em comparação com o PIC e o Arduino, seu consumo ainda é inferior ao de dispositivos como o X-Core.

2.3.2 Linguagem de programação

Para o desenvolvimento do *software* a ser executado no ESP32 foi utilizada a linguagem de programação C++. Se trata de uma linguagem de programação robusta que combina recursos de alto nível, como abstração de dados e orientação a objetos, permitindo acesso direto ao *hardware*. Esse acesso direto é essencial em sistemas embarcados, pois permite interagir diretamente com os periféricos e registradores do microcontrolador, proporcionando maior controle sobre o dispositivo (SOUZA, G. V. P. d., 2022).

Outro fator importante é que a linguagem C++ é suportada pela plataforma Arduino IDE, que facilita o desenvolvimento ao oferecer suporte nativo, bibliotecas especializadas e ferramentas que simplificam a escrita, compilação e carregamento do código para o ESP32.

2.3.3 Arduino IDE

O Arduino IDE é um *software* que oferece uma plataforma de desenvolvimento integrada para programação e gravação de códigos em microcontroladores, utilizando as linguagens C e C++ (MCROBERTS, 2018).

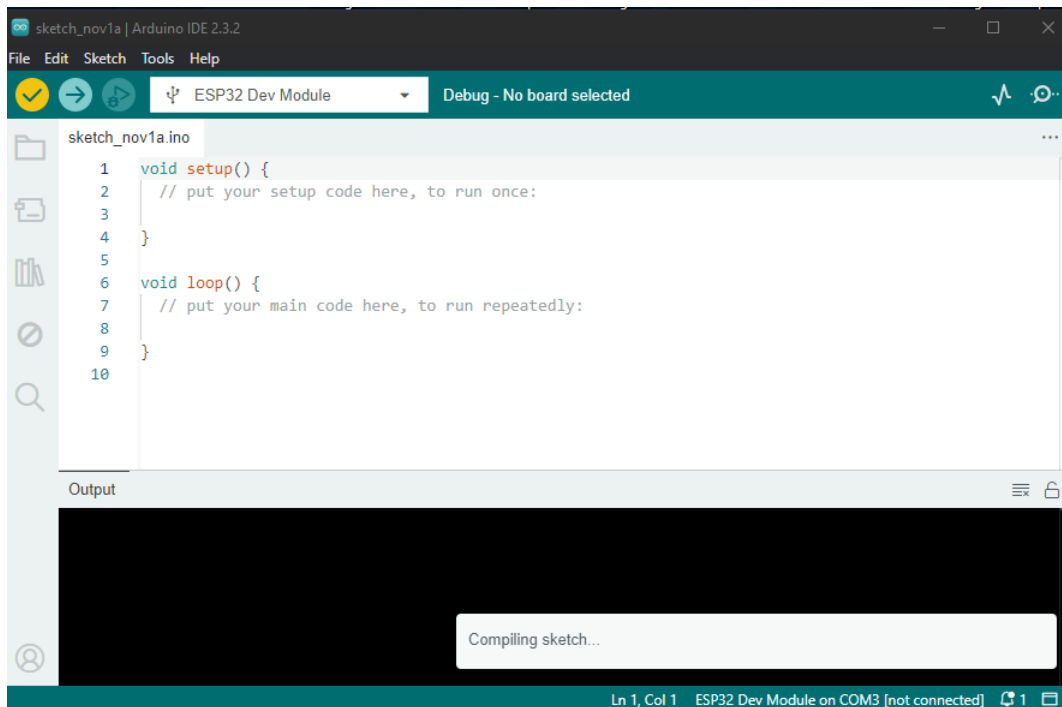
A plataforma Arduino IDE é um ambiente de desenvolvimento integrado utilizado para programar dispositivos como Arduino e ESP32, fornecendo uma interface gráfica que facilita a escrita, a compilação e o carregamento de códigos nos microcontroladores (FERREIRA; FAGUNDES, s.d.).

A interface do Arduino IDE é organizada de forma simples. Um recurso útil fornecido pela plataforma é o monitor serial, que permite a comunicação direta entre o microcontrolador e o computador. Esse recurso é fundamental para a depuração e o monitoramento em tempo real, possibilitando a visualização de dados e mensagens geradas pelo sistema em execução.

Uma característica importante da plataforma é o conceito de *sketches*, que representa o programa desenvolvido para o microcontrolador. Cada *sketch* possui duas funções principais: a função *setup*, executada uma vez no início para inicializar os pinos e realizar as configurações iniciais, e a função *loop*, que é executada continuamente em um ciclo infinito e contém a lógica principal do programa (ARDUINO.CC, 2023).

A Figura 4 ilustra a interface do Arduino IDE. Nela, consegue observar a seção de saída, que exibe mensagens de *output*, incluindo erros de compilação e informações de depuração. As funções *setup* e *loop*, citadas anteriormente, estão localizadas no centro da tela. À esquerda, encontramos o painel de funções e bibliotecas disponíveis, que facilita a navegação e a inserção de comandos no código. Acima, estão os botões de controle, que permitem compilar, carregar o código no microcontrolador e acessar as preferências do ambiente.

Figura 4 – Interface do Arduino IDE.



Fonte: Elaborado pela autora, 2024.

A IDE oferece recursos e bibliotecas pré-definidas que facilitam a interação com os periféricos dos microcontroladores, como sensores, motores, Diodo Emissor de Luz - LED (sigla do inglês *Light Emitting Diode*) e outros dispositivos eletrônicos.

2.4 Sensores

THOMAZINI descreve:

Termo "sensor" é empregado para designar dispositivos sensíveis a alguma forma de energia, que pode ser luminosa, térmica ou

cinética. O objetivo é relacionar informações sobre uma grandeza que precisa ser medida, como temperatura, pressão, velocidade, corrente, aceleração, posição, entre outras. (THOMAZINI; ALBUQUERQUE, 2020)

Os sensores fornecem informações ao circuito eletrônico sobre eventos externos que exigem uma resposta ou ação específica. São dispositivos sensíveis a diferentes formas de energia, como luminosa, térmica ou cinética, e têm o propósito de medir grandezas como temperatura, pressão e velocidade (WENDLING, M., 2010).

Devido à diversidade de modelos de sensores disponíveis, é essencial escolher o tipo mais adequado com base nas necessidades específicas do projeto. A seleção deve considerar as características do sensor e os requisitos do sistema para assegurar a precisão e a eficácia das medições.

Ao selecionar um sensor para um projeto, é fundamental avaliar as especificidades do trabalho e as condições em que o sensor será utilizado, garantindo assim a adequação e a precisão nas análises (NEWTON, 2012).

Após analisar as especificidades dos modelos de sensores disponíveis e as necessidades do projeto, optou-se pelos sensores YL-69 e DHT11. A escolha foi baseada na relação custo-benefício, na adequação às necessidades do projeto e na facilidade de integração com a plataforma utilizada.

A seguir, nas subseções 2.4.1 e 2.4.2, são apresentados os sensores escolhidos, bem como um comparativo com outros modelos considerados e as razões para a escolha final.

2.4.1 Sensor de Umidade YL-69

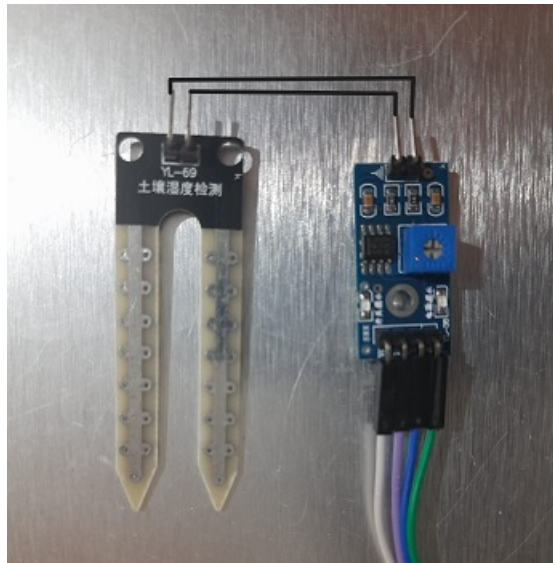
O sensor YL-69 é um dispositivo eletrônico resistivo projetado para medir a umidade do solo. Ele desempenha funções importantes em diversas aplicações (LIMA PEREIRA *et al.* 2021), incluindo agricultura, monitoramento ambiental, controle de umidade em ambientes fechados e sistemas de irrigação automatizados.

Esse sensor gera um sinal analógico com base na variação da resistência elétrica do solo, que é proporcional ao teor de umidade. Ele funciona com duas hastas metalizadas isoladas, alimentadas por uma tensão contínua (DINIZ, 2017). O sensor converte as informações de umidade em um sinal elétrico, alterando proporcionalmente a resistência entre os eletrodos.

A Figura 5 demonstra o sensor de umidade YL-69 utilizado neste projeto.

Outros modelos considerados foram o sensor FC-28 e sensores capacitivos de umidade. O FC-28 é semelhante ao YL-69, mas apresenta menor precisão em ambientes muito úmidos e maior suscetibilidade à corrosão. Sensores capacitivos, por outro lado, oferecem maior durabilidade e precisão, porém a um custo significativamente mais elevado.

Figura 5 – Sensor de Umidade YL-69.



Fonte: Elaborado pela autora, 2024.

O YL-69 foi escolhido devido ao seu bom custo-benefício, precisão satisfatória e facilidade de integração ao sistema, apresentando um desempenho adequado às necessidades do projeto a um custo relativamente baixo.

A Tabela 4 apresenta as especificações e pinagens do sensor de umidade YL-69.

Tabela 4 – Especificações e Pinagem do sensor YL-69.

Especificações e pinagem	
Tensão de Operação	3,3-5V
Sensibilidade ajustável via potenciômetro	Sim
Saída Digital	Sim
Saída Analógica	Sim
D0	Saída Digital
A0	Saída Analógica

Fonte: Adaptado de (DINIZ, 2017).

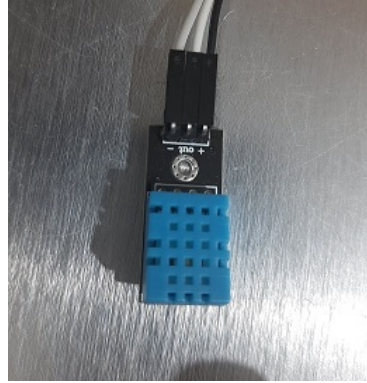
2.4.2 Sensor de Temperatura e Umidade do Solo DHT11

O sensor de temperatura e umidade do solo DHT11, ilustrado na Figura 6, é projetado para medir tanto a temperatura quanto a umidade relativa do ambiente em que está instalado. Este sensor é composto por um termistor para a medição da temperatura e por um elemento capacitivo para a medição da umidade.

O funcionamento do sensor baseia-se na conversão dessas grandezas físicas em sinais elétricos. Quando o sensor é exposto a variações de temperatura e umidade, o termistor e o elemento capacitivo alteram sua resistência elétrica e capacitância, res-

pectivamente. Essas alterações são convertidas em sinais digitais pelo próprio sensor e transmitidas aos controladores (SAXENA, 2018).

Figura 6 – Sensor de Temperatura e Umidade DHT11.



Fonte: Elaborado pela autora, 2024.

Outro modelo de sensor de temperatura e umidade do ar considerado foi o DHT22. Este modelo oferece maior precisão e um intervalo de medição mais amplo em comparação com o DHT11. No entanto, o DHT11 foi escolhido devido ao seu custo acessível e a sua facilidade de uso para a faixa de precisão exigida pelo projeto. Embora sua precisão seja inferior à do DHT22, o DHT11 satisfaz adequadamente as necessidades básicas de medição de temperatura e umidade do ar. As especificações do DHT11 pode ser vista na Tabela 5.

Tabela 5 – Especificações do Sensor de Temperatura e Umidade DHT11.

Especificações	
Tensão de Operação	3,3-5V
Tipo do Sensor	Resistor Polimérico
Faixa de Umidade	20 a 90% RH
Faixa de Temperatura	0º a 50ºC

Fonte: Adaptado de (SAXENA, 2018).

2.5 Relé

O relé é um dispositivo de acionamento eletromecânico que atua como um interruptor controlado por sinal elétrico (BRAGA, 2017). Foram utilizados dois modelos diferentes de relés para a execução deste trabalho.

O relé é acionado por um contato elétrico, utilizando-se de uma força mecânica em resposta a um estímulo (RUSH, 2011). Esse estímulo é originado por um sinal de comando vindo de um controlador. De acordo com (STROSKI, 2018), a força mecânica do

atuador opera o contato elétrico por um eletroímã que recebe corrente elétrica e produz um campo magnético, atraindo um pedaço metálico ligado ao contato móvel.

O modelo utilizado no primeiro circuito elétrico é o relé modular, que foi conectado diretamente à bomba d'água. Ele faz parte do circuito de controle, sendo responsável por ativar e desativar a bomba, de acordo com os sinais recebidos do sistema de controle. Além disso, o relé está conectado à alimentação elétrica e ao X-Core.

O relé modular é empregado em instalações elétricas fixas, como sistemas residenciais, industriais ou comerciais. Projetado para operar em ambientes de alta tensão, ele é robusto, garantindo a ativação precisa de componentes que requerem maior potência, como a bomba d'água neste projeto.

Uma representação visual do relé pode ser observada na Figura 7.

Figura 7 – Relé modular utilizado no projeto.



Fonte: Elaborado pela autora, 2024.

O segundo modelo de relé é integrado à uma placa de circuito impresso - PCB (sigla do inglês *printed circuit board*). O módulo relé contém quatro relés integrados. Esses relés são versáteis e frequentemente utilizados em projetos eletrônicos e de automação, situação em que relés individuais são necessários para controlar dispositivos específicos. Eles são conectados a sistemas controlados por microcontroladores e podem ser facilmente adaptados para diferentes aplicações.

A Figura 8 representa o módulo relé utilizado na placa.

No projeto, foram utilizados dois tipos de relés. O primeiro, um relé modular, foi empregado para o sistema X-Core, possibilitando o acionamento da bomba, já que o sistema original não tinha capacidade para operar diretamente em 220V, limitando-se a saídas de 24V. O segundo, um módulo relé, que faz parte do sistema atual, foi integrado ao sistema desenvolvido, eliminando a necessidade do primeiro relé modular. O novo módulo relé é capaz de funcionar com a voltagem necessária, permitindo a integração de até quatro dispositivos simultaneamente (PEREIRA, F., s.d.).

Figura 8 – Módulo relé utilizado no projeto.



Fonte: Elaborado pela autora, 2024.

2.6 Módulo RTC

O Módulo RTC DS1307 (Figura 9) é um relógio de tempo real de alta precisão e baixo consumo de energia, projetado para manter a contagem precisa de tempo em projetos eletrônicos. Ele é utilizado em aplicações que requerem sincronização de horário e data, como relógios digitais, sistemas de registro de dados e dispositivos de controle de acesso (RAVI, 2022).

O RTC possui uma memória não volátil capaz de armazenar informações sobre horas, minutos, segundos, dia, mês e ano. Uma de suas principais características é a capacidade de manter esses dados mesmo em caso de interrupção no fornecimento de energia principal. Isso ocorre devido à existência de uma bateria interna que mantém o RTC alimentado.

Figura 9 – Módulo RTC utilizado no projeto.



Fonte: Elaborado pela autora, 2024.

O módulo RTC DS1307 se comunica com microcontroladores e microprocessadores através do barramento I2C e geralmente inclui uma EEPROM 24C32 (*Electrically Erasable Programmable Read-Only Memory*), também conectada ao mesmo barramento. Esta EEPROM proporciona armazenamento adicional de 32 KB para dados relevantes

ao projeto, permitindo a preservação de informações importantes, como configurações e registros históricos, independentemente da memória do RTC.

O ESP32 possui um RTC interno, no entanto, o *clock* desse RTC pode apresentar uma variação de cerca de 5%, o que pode resultar em discrepâncias ao longo do tempo. Por esse motivo, a melhor escolha foi acrescentar o módulo RTC DS1307, que garante maior precisão na contagem de tempo e assegura a continuidade das funções temporais do sistema.

A Tabela 6 apresenta as principais características do módulo RTC DS1307, destacando sua comunicação I2C e a memória não volátil de 56 *bytes*. Essa memória é fundamental para manter a contagem precisa de tempo mesmo na ausência de energia, garantindo a continuidade das funções temporais do sistema.

Tabela 6 – Especificações do Módulo RTC DS1307 + EEPROM 24C32 I2C.

Especificações	
<i>Chip</i>	DS1307
Tensão	3,3-5V
Memória	56 bytes de memória não volátil
Armazenamento de memória	24C32 EEPROM I2C 32
Comunicação	I2C
Bateria compatível	CR2032

Fonte: Adaptado de informações fornecidas pelo fabricante.

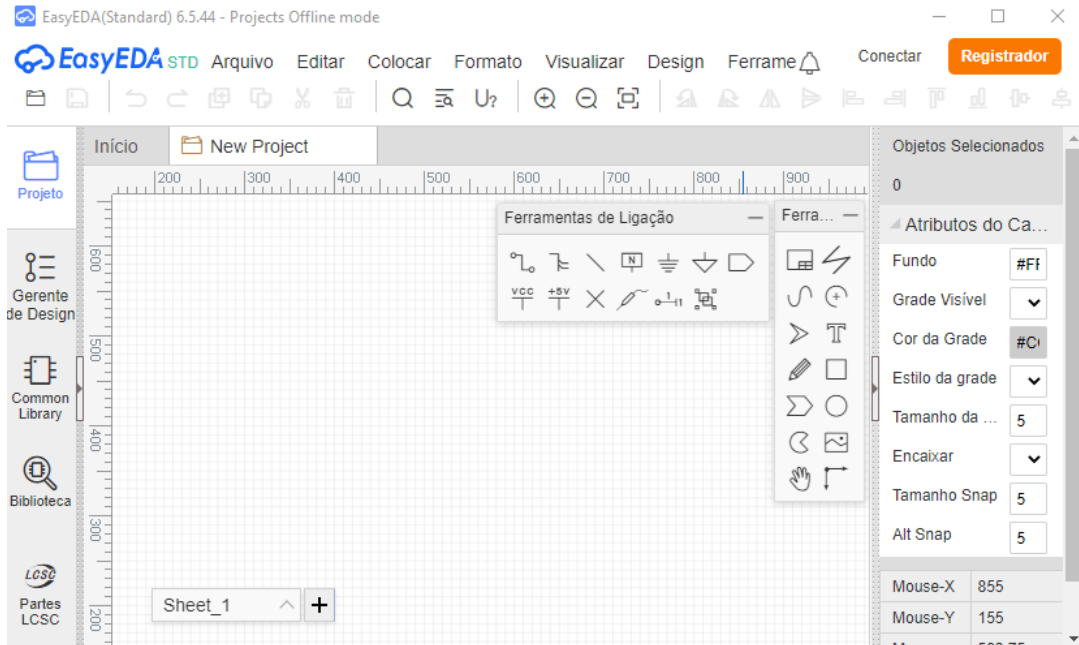
2.7 EasyEda

O EasyEDA é uma ferramenta de EDA (*Electronic Design Automation*) que fornece uma gama de recursos para projetar, simular e compartilhar esquemas, simulações e layouts de PCBs. Este *software* versátil permite a criação e edição de diagramas esquemáticos, além de oferecer uma ferramenta de simulação SPICE (Programa de Simulação com Ênfase em Circuito Integrado) para analisar o comportamento de circuitos analógicos e digitais (CIRCUITSTODAY, 2020).

Com o EasyEDA, os usuários podem selecionar componentes da biblioteca integrada, conectá-los para representar a lógica do circuito, posicionar os componentes no PCB, traçar as trilhas de conexão, definir as camadas do PCB, simular o desempenho do circuito e verificar se ele atende aos requisitos do projeto.

A Figura 10 mostra a interface do EasyEDA, ilustrando as principais funcionalidades disponíveis para o desenvolvimento e análise de projetos eletrônicos.

Figura 10 – Interface do *software* EasyEDA.



Fonte: Elaborado pela autora, 2024.

2.8 Estado da arte

Nos últimos anos, diversos estudos têm sido realizados com o objetivo de otimizar o processo de irrigação de estufas, por meio do uso de microcontroladores, com vistas a garantir um ambiente adequado para o desenvolvimento saudável das plantas.

Um destes trabalhos na área de irrigação foi realizado por Marques Filho (2017), que investigou o uso de um sistema microcontrolado para a promover a irrigação. O estudo demonstrou que a utilização do referido sistema resultou em ganhos significativos para a produção das mudas, proporcionando o aumento da produtividade e promovendo o desenvolvimento de plantas mais saudáveis.

O sistema microcontrolado utilizou a plataforma Arduino Mega. Foram implantados sensores de umidade do solo para monitorar continuamente as condições de umidade e, como consequência, acionar o sistema de irrigação quando fosse necessário. Por intermédio do microcontrolador foi possível automatizar o processo de irrigação, de modo a garantir a dispersão adequada da água e evitar o desperdício.

O estudo realizado por Henriques *et al.* (2021) demonstrou a eficiência do sistema de irrigação automatizado, especialmente quando a irrigação foi acionada com base em condições de umidade do solo predefinidas. O emprego do microcontrolador ESP32 resultou em um controle preciso do processo de irrigação, tendo sido destacada a importância do uso de tecnologias microcontroladas para otimizar a irrigação.

Sousa, Silva Nascimento e Marcos Pereira dos Santos (2022) também desenvolveu um sistema microcontrolado para a irrigação, utilizando inicialmente o Arduino Uno.

No entanto, devido às limitações do Arduino Uno em termos de conectividade e capacidade de processamento, houve a necessidade de troca para o microcontrolador ESP8266. Neste estudo, as informações obtidas por meio do sensor de temperatura foram disponibilizadas em um *website*, podendo ser monitoradas à distância.

Fernandes (2017) desenvolveu um Sistema Automatizado de Controle de Estufas para Cultivo de Hortaliças que teve por base a implementação de um sistema de controle das condições ambientais de estufas. O sistema é baseado em Arduino e controlado por um aplicativo *Android*, que armazena os dados em uma base de dados MySQL, realizando o monitoramento da temperatura, da luminosidade e da umidade do solo no local.

Levando em consideração especificamente os sistemas microcontrolados de irrigação, é possível afirmar que seu uso pode promover um aumento na produtividade agrícola, além de proporcionar economia de água e redução de mão de obra. Esses sistemas contribuem para a otimização do processo produtivo e, em última análise, promovem a sustentabilidade ambiental ao utilizar recursos de forma mais eficiente e reduzir o desperdício.

As especificações de cada trabalho são apresentadas no Quadro 2, que compara os componentes utilizados e outras características relevantes.

Diferentemente dos sistemas anteriores que utilizaram o Arduino Mega, o presente estudo optou pelo uso do ESP32. Este microcontrolador facilita e promove, de forma direta, a comunicação entre o usuário do sistema e as informações fornecidas pelos atuadores, sensores e periféricos, devido à sua conectividade integrada. Isso representa uma vantagem, pois reduz a necessidade de serem utilizados componentes adicionais para estabelecer conectividade.

Comparado ao sistema desenvolvido por Henriques *et al.* (2021), que também utilizou o ESP32 como microcontrolador, o presente estudo se destaca pela inclusão de um módulo RTC. O uso desse equipamento é importante em caso de queda de energia, pois permite que o sistema mantenha o controle das horas programadas para a irrigação sem a necessidade de intervenção manual.

Além disso, o sistema desenvolvido neste estudo foi implementado diretamente na estrutura real da estufa existente na instituição de ensino IFMG-*Campus Bambuí*. Diferentemente de estudos anteriores que foram desenvolvidos em protótipos ou maquetes em pequena escala, este projeto foi realizado em condições de uso real.

É importante reconhecer, no entanto, que uma limitação do sistema apresentado é que ele não controla diretamente a irrigação com base nas variáveis monitoradas, como umidade do solo e temperatura. Atualmente, o sistema se limita a monitorar essas variáveis e exibir as informações para o usuário. Esse aspecto pode ser abordado em trabalhos futuros, com a implementação de um controle automatizado da irrigação baseado nos dados coletados.

O sistema desenvolvido possui flexibilidade, não sendo um modelo fechado. Ele

Quadro 2 – Comparação entre os trabalhos utilizados como referência para este projeto.

	Fernandes et al. (2017)	Marques (2017)	Henriques (2021)	Sousa (2022)	Presente Trabalho
Microcontrolador	Arduino Mega	Arduino Mega	ESP32	Arduino Uno/ESP8266	ESP32
Sensores Utilizados	Umidade do Solo, Temperatura, Luminosidade	Temperatura, umidade do solo e iluminação	Temperatura e Umidade do solo	Umidade do Solo, Temperatura	Umidade do Solo, Temperatura e Umidade do ar
Conectividade	Wi-Fi	Não especificado	Wi-Fi	Wi-Fi	Wi-Fi
Timer automático	Sim	Sim	Não	Sim	Sim
Interface de Controle	Aplicativo Android	Não especificado	Website	Website	Website
Implementação de uma base de dados	Sim	Não	Não especificado	Não	Não
Implementação	Simulação/protótipo	protótipo instalado	Simulação/protótipo	protótipo instalado	protótipo instalado
expansível	Não especificado	Não especificado	Não especificado	Sim	Sim

Fonte: Elaborado pela autora, 2024.

permite a instalação de novos equipamentos na placa, com a possibilidade de adicionar ou retirar saídas, facilitando a expansão do sistema conforme as necessidades futuras.

3 METODOLOGIA

Este capítulo descreve a metodologia adotada para o desenvolvimento e implementação do projeto de irrigação automatizada, seguindo os princípios da DSR. Essa abordagem orienta a criação e a avaliação de artefatos tecnológicos voltados para a solução de problemas práticos em contextos reais.

Na Seção 3.1 é discutida a classificação do trabalho dentro da metodologia DSR, detalhando como o projeto se enquadra nesse modelo e os critérios utilizados para sua avaliação.

A Seção 3.2 descreve a solução implementada, detalhando a abordagem adotada para resolver o problema identificado e a estruturação da proposta. A Subseção 3.2.1 oferece uma visão geral dos materiais, equipamentos e tecnologias utilizados no projeto. Já a Subseção 3.2.2 apresenta uma análise dos custos envolvidos, proporcionando uma visão dos recursos financeiros necessários para sua execução.

3.1 Classificação do trabalho

A metodologia deste projeto de pesquisa se classifica, conforme Gerhardt e Silveira (2009), da seguinte maneira: quanto à abordagem, trata-se de uma pesquisa qualitativa, já que foca na análise da qualidade do produto proposto; em relação à natureza, é uma pesquisa aplicada, pois busca a aplicação prática de conhecimentos científicos e tecnológicos. No que diz respeito aos objetivos, caracteriza-se como uma pesquisa exploratória, pois visa formular hipóteses, identificar soluções e estabelecer uma base sólida de conhecimento. Quanto aos procedimentos, enquadra-se no aprimoramento de ideias, com foco no desenvolvimento e aperfeiçoamento do sistema.

Sob a perspectiva da DSR, o projeto adota um ciclo iterativo de desenvolvimento, no qual as etapas de concepção, implementação, avaliação e refinamento são repetidas até que o sistema atenda plenamente aos requisitos e objetivos estabelecidos. O artefato desenvolvido, conforme Lacerda *et al.* (2013), trata-se de uma instância de método, implementando tanto um protótipo quanto um sistema funcional, culminando em sua aplicação no ambiente real. Este trabalho também pode ser caracterizado como uma pesquisa-ação, por seu enfoque na identificação, análise e solução de problemas práticos.

De acordo com a classificação de Wazlawick (2009), este projeto de pesquisa se enquadra na modalidade de *Apresentação de algo Presumivelmente Melhor*, uma vez que visa desenvolver um sistema de irrigação automatizado mais eficiente e sustentável em comparação ao sistema atualmente empregado no viveiro do IFMG - Campus Bambuí.

3.2 Solução Proposta

A solução desenvolvida para o problema identificado consiste na criação de um protótipo de sistema para o monitoramento e controle automatizado da irrigação no viveiro do IFMG - *Campus* Bambuí. O microcontrolador ESP32 foi escolhido para gerenciar o controle do sistema e realizar a comunicação via rede *Wi-Fi*.

A configuração remota é realizada por meio de uma interface *web*, construída em *Hypertext Markup Language* (HTML), que é hospedada no próprio ESP32. Essa interface é acessada através da rede local pelo endereço de *Internet Protocol* (IP) do dispositivo, utilizando o protocolo *Hypertext Transfer Protocol* (HTTP) para a comunicação cliente-servidor.

A interface *web* permite ao usuário ajustar parâmetros do sistema, como o horário e a duração da irrigação, além de possibilitar o controle manual para ativação ou desativação da irrigação, proporcionando maior flexibilidade operacional. Também é possível monitorar, em tempo real, a temperatura ambiente e os níveis de umidade do ar e do solo.

O sistema realiza monitoramento contínuo, com os sensores de umidade do solo (YL-69, temperatura e umidade do ar (DHT11) atualizando as informações em intervalos de um segundo. A irrigação é acionada de acordo com horários predefinidos, ativando os relés que controlam as válvulas de irrigação por uma duração determinada. Para garantir precisão nos horários, o módulo RTC DS1307 foi integrado ao sistema.

A Figura 11 ilustra a estrutura do viveiro do IFMG - *Campus* Bambuí, onde o sistema de irrigação automatizado foi implementado. À direita, observa-se uma sala de controle que abriga os dispositivos e equipamentos responsáveis pela automação do processo de irrigação. Neste espaço, foi instalado o protótipo desenvolvido para o projeto.

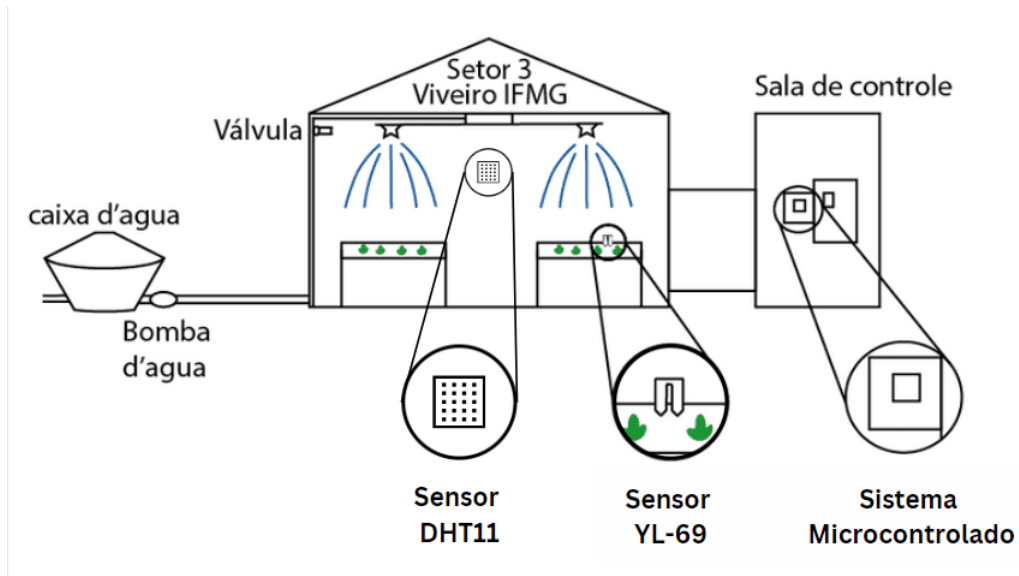
O sistema inclui um sensor de umidade (YL-69) inserido no solo, e um sensor de temperatura e umidade (DHT11) instalado no ambiente. A caixa d'água e a bomba, visíveis na parte esquerda da Figura, são responsáveis pelo armazenamento e distribuição da água, enquanto uma válvula controla o fluxo para os aspersores. Tanto a bomba quanto a válvula são controladas automaticamente pelo sistema, conforme os parâmetros definidos.

O diagrama de casos de uso, representado na Figura 12, ilustra as principais funcionalidades e interações entre o usuário e o sistema de irrigação desenvolvido. Ele descreve as operações que podem ser realizadas e como as ações do usuário se integram às tarefas executadas pelo ESP32.

O sistema possui dois atores principais: o usuário e o ESP32. O usuário interage com o sistema, configurando e monitorando os processos de irrigação, enquanto o ESP32 atua como controlador, recebendo comandos e gerenciando as operações automatizadas.

As interações do usuário com o sistema incluem: a ativação e desativação da irrigação, onde o usuário pode controlar manualmente o sistema; a configuração de horários de irrigação, permitindo a programação de horários específicos; e o monitoramento tanto

Figura 11 – Fluxograma da estrutura do viveiro IFMG.

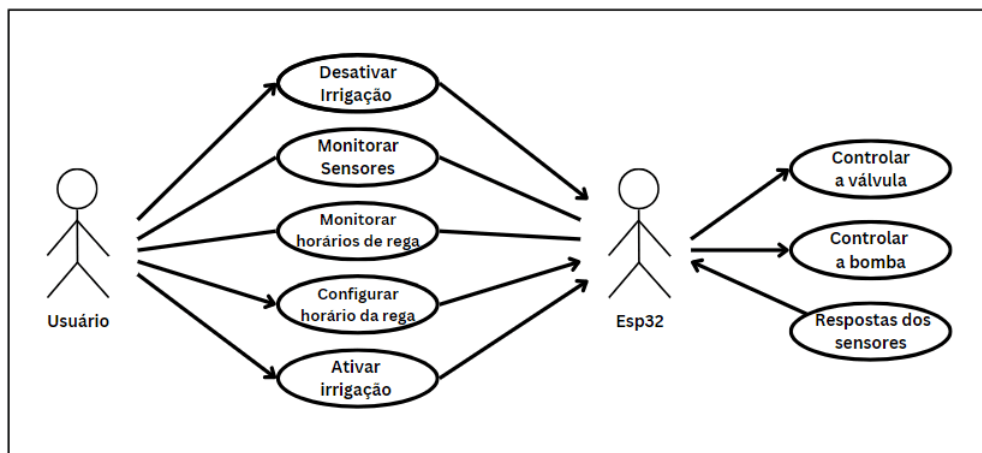


Fonte: Elaborado pela autora, 2024.

dos horários quanto dos sensores, verificando se o sistema opera conforme o planejado, além de disponibilizar uma visão em tempo real das condições do ambiente.

O ESP32, por sua vez, executa as operações conforme os comandos recebidos, controlando a abertura e fechamento da válvula de irrigação e da bomba d'água. Ele também processa as leituras dos sensores, fornecendo ao usuário uma resposta sobre as condições monitoradas.

Figura 12 – Diagrama de caso de uso do sistema desenvolvido.



Fonte: Elaborado pela autora, 2024.

3.2.1 *Materiais, Equipamentos e Tecnologias*

Nesta Subseção são detalhados os materiais, equipamentos e tecnologias empregados no desenvolvimento e implementação do projeto. Os itens são organizados em três categorias principais:

1. **Materiais Utilizados na Confeção do protótipo:** Incluem todos os materiais necessários para a montagem física da placa e a acomodação da mesma no protótipo final.
2. **Equipamentos contidos no viveiro e instalados na placa:** Refere-se aos equipamentos presentes no viveiro e os utilizados no protótipo.
3. **Tecnologias Empregadas no Desenvolvimento do Projeto:** Abrange todas as tecnologias e bibliotecas utilizadas no código, além dos softwares usados na criação e simulação dos códigos e protótipo final.

A seguir, cada uma dessas categorias são detalhadas.

1. **Materiais:**

a) **Materiais utilizados no desenvolvimento:**

- i. **Solda:** Material utilizado para fixar os componentes na placa.
- ii. **Fios/*Jumpers*:** Utilizados para realizar conexões temporárias e permanentes entre os componentes.
- iii. **Placa de Fenolite Cobreada:** Placa usada para impressão do circuito impresso e fixação dos componentes eletrônicos.
- iv. **Fita Crepe:** Aplicada para organizar os fios.
- v. **Caixinha de Madeira com Tampa:** Usada para guardar os equipamentos e componentes.
- vi. **Ferro de solda:** Utilizado para soldar os componentes eletrônicos na placa de fenolite.

2. **Equipamentos:**

a) **Equipamentos do viveiro:**

- i. **Válvula de Irrigação:** Controla o fluxo de água para os aspersores.
- ii. **Bomba de Água:** Responsável por bombear a água da caixa d'água para os aspersores.
- iii. **Caixa d'Água:** Armazena a água utilizada para a irrigação.
- iv. **Aspersores:** Dispositivo usado para irrigar as plantas na forma de chuva artificial.

- v. **Canos:** Tubulações para o transporte da água.
- vi. **Transformador:** Responsável por ajustar a tensão elétrica adequada aos equipamentos.

b) **Equipamentos utilizados na placa:**

- i. **Microcontrolador ESP32:** Responsável pelo controle geral do sistema.
- ii. **Sensor de Umidade do Solo (YL-69):** Utilizado para monitorar o nível de umidade do solo.
- iii. **Sensor de Temperatura e Umidade (DHT11):** Utilizado para monitorar a temperatura ambiente e a umidade do ar.
- iv. **RTC DS1307:** Módulo de relógio em tempo real para garantir a precisão dos horários de irrigação.
- v. **Relés:** Utilizados para controlar a abertura e fechamento da válvula de irrigação e da bomba d'água.
- vi. **Fonte de Alimentação de 5V:** Fornece energia elétrica para o funcionamento do sistema.

3. Tecnologias:

a) **Tecnologias utilizadas no código:**

- i. **HTML:** Utilizados para a estrutura da interface *web*.
- ii. **CSS:** Utilizados para a estilização da interface *web*.
- iii. **HTTP:** Protocolo utilizado para a comunicação entre o servidor (ESP32) e os clientes (usuários).
- iv. **C++:** Linguagem usada para programar o microcontrolador.

b) **Bibliotecas:**

- i. **RTClib:** Biblioteca para interação com o módulo RTC para leitura de data e hora.
- ii. **Arduino.h:** Biblioteca que oferece funções básicas de controle de *hardware*.
- iii. **Wire.h:** Biblioteca utilizada para a comunicação I2C entre o microcontrolador e o módulo RTC.
- iv. **DHT:** Biblioteca para interagir com o sensor DHT11.
- v. **Wi-Fi:** Biblioteca usada para comunicação sem fio utilizada para conectar o ESP32 à rede local.
- vi. **WebServer:** Biblioteca para criar um servidor *web* no ESP32.

c) **Softwares:**

- i. **Arduino IDE:** Ambiente de desenvolvimento integrado para programar o microcontrolador ESP32.

- ii. **EasyEDA**: Utilizado para desenvolver o circuito impresso.
- iii. **Wokwi**: Utilizado no início para desenvolver o projeto da placa e do código.
- iv. **CadeSimu**: Utilizado para desenvolver o projeto elétrico do viveiro.
- v. **Adobe Illustrator**: Utilizado para criação dos fluxogramas e diagramas contidos no texto.

3.2.2 *Orçamentos*

Os materiais e equipamentos necessários para a execução do projeto foram adquiridos em uma loja especializada. A lista inclui o microcontrolador ESP32, os sensores de umidade e temperatura, o módulo RTC DS1307, o relé, a solda de estanho, os fios de *jumper* e a placa de fenolite.

Os demais itens, como a fonte de 5V, fita crepe, caixinha de madeira e ferro de solda, já estavam disponíveis no laboratório. O *notebook* utilizado para estudos, programação e redação da monografia também era propriedade da discente.

A Tabela 7 apresenta os valores detalhados dos materiais e equipamentos adquiridos, bem como aqueles que estavam disponíveis no laboratório.

Tabela 7 – Orçamento dos Materiais e Equipamentos.

Item	Quantidade	Preço Unitário (R\$)	Total (R\$)
ESP32	1	39,90	39,90
Sensor (YL-69)	1	13,99	13,99
Sensor (DHT11)	1	12,90	12,90
RTC DS1307	1	20,00	20,00
Relés	1	24,99	24,99
Fonte de 5V	1	0	0
Solda estanho	1	12,99	12,99
Fios de <i>Jumper</i>	20	0,50	10,00
Placa de Fenolite Cobreada (15cmX15cm)	1	20,99	20,99
Fita Crepe	1	0	0
Caixinha de Madeira	1	0	0
Ferro de solda	1	0	0
Total Geral			154,76

Fonte: Elaborado pela autora, 2024.

Após a apresentação da metodologia utilizada para o desenvolvimento do sistema, o Capítulo 4 se concentrará no desenvolvimento prático do projeto. Serão explicadas as etapas de implementação, e detalhadas as fases de inspeção e reparo dos equipamentos, além da avaliação do X-Core.

4 DESENVOLVIMENTO

O desenvolvimento deste projeto ocorreu em três fases distintas. Na primeira etapa, foi realizada uma revisão bibliográfica sobre projetos similares ao sistema de irrigação automatizado em desenvolvimento. Os detalhes dessas análises foram discutidos no Capítulo 2.

A segunda fase será descrito neste Capítulo, descrevendo o processo de inspeção e reparo dos equipamentos, como a substituição de uma válvula de irrigação, revisão da bomba d'água, aspersores, canos e a caixa d'água. A rede elétrica também exigiu ajustes, incluindo a troca de fios e a instalação de novos componentes. Houve também uma avaliação do X-Core, equipamento que já havia no local.

Por fim, a terceira fase etapa envolveu o desenvolvimento do protótipo, a instalação do sistema no viveiro e a verificação do seu funcionamento. O processo de instalação incluiu a conexão do protótipo ao sistema elétrico do local, a integração com a internet local e a instalação do sensor no solo, além de verificar se os dispositivos estavam sendo acionados conforme o esperado. Essa fase será detalhada no Capítulo 5, dedicado aos resultados.

4.1 Identificação e reparo dos equipamentos existentes

Foi realizada uma visita técnica ao viveiro do campus para avaliar as necessidades do espaço, compreender as demandas dos responsáveis pela operação do local e identificar as deficiências do sistema de irrigação. No Setor 3 do viveiro, havia equipamentos previamente instalados, muitos dos quais estavam danificados ou fora de operação, o que exigiu a solicitação de reparos aos responsáveis técnicos.

A Figura 13 mostra o estado da válvula de irrigação antes e depois do conserto. Durante as primeiras visitas ao viveiro, a válvula estava quebrada (Figura 13(a)). Esse equipamento é responsável por controlar o fluxo de água, e, devido ao dano, a água não alcançava os aspersores. Após a inspeção e a solicitação de reparos, a válvula foi restaurada, conforme mostrado na Figura 13(b).

Além disso, foram encontrados alguns componentes no local (Figura 14), entre eles temos uma bomba d'água (Figura 14(a)), responsável por bombear a água da caixa d'água até a válvula. Embora estivesse em pleno funcionamento, foi necessário refazer as conexões elétricas.

O viveiro contava com aspersores, responsáveis pela distribuição de água sobre as plantas. No entanto, foi constatado que vários deles estavam entupidos ou ausentes, necessitando de reparos. A Figura 14(b) ilustra uma dessas avarias, que resultava em uma distribuição irregular da água, com algumas plantas recebendo irrigação excessiva, enquanto outras permaneciam sem água.

Figura 13 – Antes e depois da válvula de irrigação instalada no viveiro.



(a) Válvula quebrada antes de arrumar.



(b) Válvula depois de arrumar.

Fonte: Elaborado pela autora, 2024.

Também foi identificada no local uma caixa d'água, responsável por armazenar a água utilizada na irrigação. A estrutura apresentava danos que exigiram reparos para evitar vazamentos. A Figura 14(c) ilustra o conserto realizado.

Figura 14 – Componentes do sistema de irrigação.



(a) Bomba d'água conectada à caixa d'água.



(b) Aspersores quebrados.



(c) Caixa d'água após fazer o reparo.

Fonte: Elaborado pela autora, 2024.

Por fim, foram identificados canos de tubulação responsáveis por conduzir a

água até os aspersores. Constatou-se que algumas seções estavam quebradas ou avariadas, comprometendo o funcionamento do sistema. Foi necessário substituir as partes danificadas para restabelecer a operação adequada.

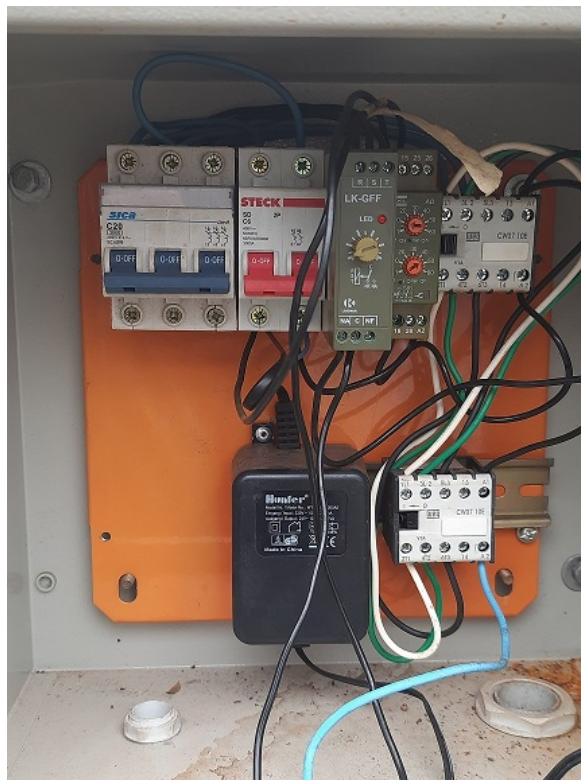
4.2 Identificação e reparo da rede elétrica

Durante a revisão e manutenção dos equipamentos do viveiro, foi necessária uma análise da rede elétrica existente. A infraestrutura, que conecta a sala de controle aos diversos dispositivos do viveiro, apresentava falhas que comprometiam o funcionamento do sistema.

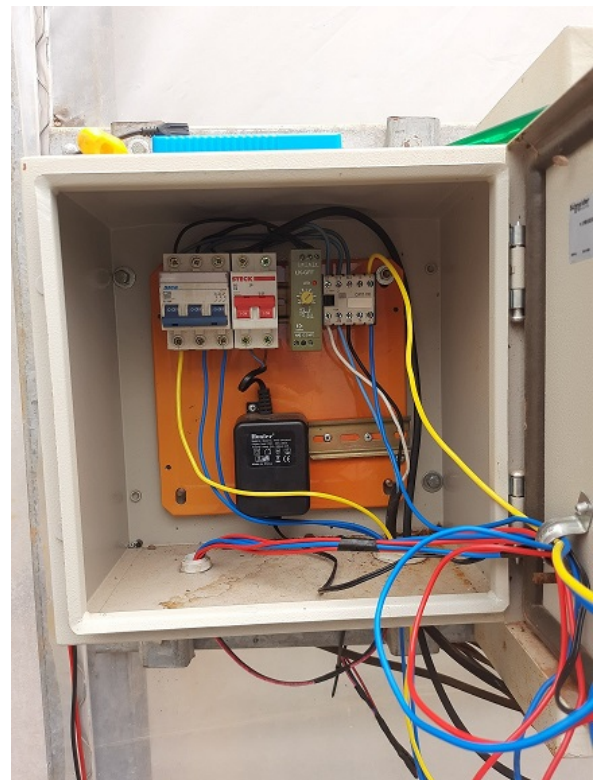
Identificou-se que o transformador, responsável por ajustar a tensão elétrica para o correto funcionamento dos equipamentos, estava defeituoso. Além disso, foi necessário refazer a instalação elétrica do viveiro, substituindo fios e corrigindo as conexões inadequadas.

A Figura 15 ilustra o processo de substituição dos fios e a reconfiguração da rede elétrica. Na Figura 15(a), observa-se os equipamentos sem todas as conexões elétricas, enquanto a Figura 15(b) mostra as ligações restauradas e organizadas, facilitando a compreensão do sistema elétrico. Após essas intervenções, a rede elétrica foi restabelecida.

Figura 15 – Equipamentos antes e depois da troca dos fios.



(a) Equipamentos sem conectar todos os fios.



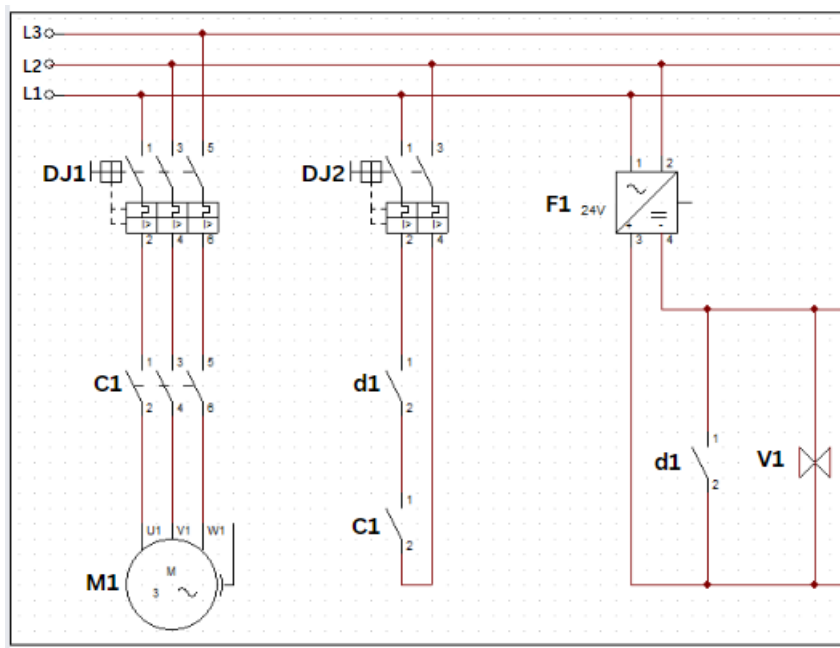
(b) Equipamentos depois de conectar todos os fios.

Na Figura 16, é possível visualizar os componentes do circuito e suas respectivas conexões. Os disjuntores DJ1 e DJ2 têm a função de proteger o sistema contra sobrecorrentes. O DJ1 está alocado para proteger o circuito do motor (M1), enquanto o DJ2 protege o circuito que alimenta o X-Core e a válvula de irrigação (V1).

As linhas de alimentação L1, L2 e L3 representam as fases de uma rede trifásica, correspondendo às fases R, S e T, respectivamente. O motor M1, responsável pelo acionamento da bomba d'água, é controlado pelo contator C1. A ativação do C1 ocorre por meio do relé modular (d1), que integra o sistema. Quando o relé d1 é acionado, o contator C1 é energizado, permitindo que o motor M1 entre em operação. Foi necessária utilização do relé modular neste circuito, para possibilitar o acionamento da bomba em 220V, uma vez que o relé simples proveniente do X-Core opera apenas em 24V, o que seria insuficiente para o funcionamento adequado do motor.

O X-Core, representado pelo F1, atua como controlador central do sistema de irrigação, sendo equipado com um transformador interno que realiza a conversão de 220V para 24V, tensão utilizada para controlar a válvula de irrigação (V1). O relé interno do X-Core é o responsável por acionar a válvula. Quando o X-Core realiza a abertura da válvula, o fluxo de água é liberado.

Figura 16 – Esquema elétrico da rede com o X-Core.



Fonte: Elaborado pela autora, 2024.

A Figura 17 refere-se ao diagrama do circuito elétrico da instalação associada ao protótipo de irrigação desenvolvido.

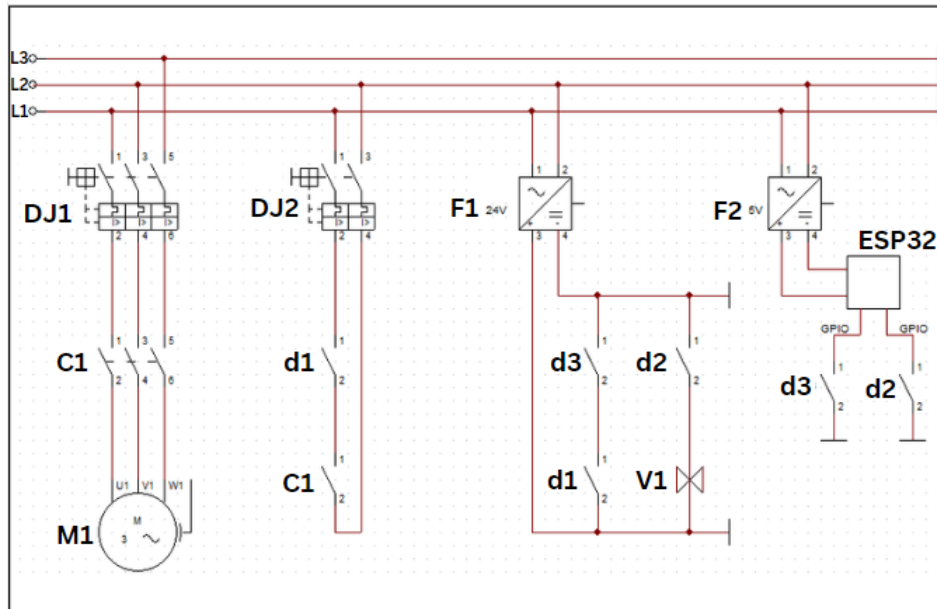
Semelhante ao circuito anterior, os disjuntores DJ1 e DJ2 protegem, respectivamente, o circuito da bomba e o sistema de controle. Os condutores L1, L2 e L3 representam

as fases de uma rede trifásica, correspondendo às fases R, S e T. Os componentes M1 (motor da bomba), C1 (contator) e d1 (relé modular) mantiveram-se inalterados em relação ao esquema anterior.

Adicionalmente, foi instalada uma fonte de alimentação representada por F2, a qual converte a tensão de 220V para 5V. Essa fonte é responsável por alimentar o protótipo desenvolvido. No projeto, estão integrados dois relés, d2 e d3, que controlam, respectivamente, a bomba e a válvula de irrigação.

Devido à saída máxima de 3,3V do ESP32, foi necessário incluir um regulador de tensão, representado por F1, para converter a tensão de 220V para 24V e assim acionar a válvula de irrigação (V1), substituindo a fonte interna utilizada pelo controlador X-Core no projeto anterior. O relé modular d1 continua presente no sistema, enquanto os relés d2 (relé 2) e d3 (relé 3) controlam a válvula e a bomba de água, respectivamente.

Figura 17 – Esquema elétrico da rede com a substituição do X-Core pelo protótipo.



Fonte: Elaborado pela autora, 2024.

4.3 Inspeção no X-core

Outro desafio enfrentado foi relacionado ao X-Core, equipamento inicialmente previsto para ser utilizado no projeto com o objetivo de automatizar a irrigação durante o desenvolvimento do protótipo. Sua implementação visava reduzir a carga de trabalho dos responsáveis pelo viveiro, eliminando a necessidade de rega manual.

Durante os testes em laboratório, constatou-se que o X-Core apresentava falhas nas fiações internas. Com o auxílio do técnico laboratorista Reginaldo, o problema foi solucionado e o equipamento foi reinstalado no viveiro. No entanto, os problemas com

outros dispositivos do viveiro ainda não haviam sido completamente resolvidos, o que impossibilitou a instalação definitiva do X-Core.

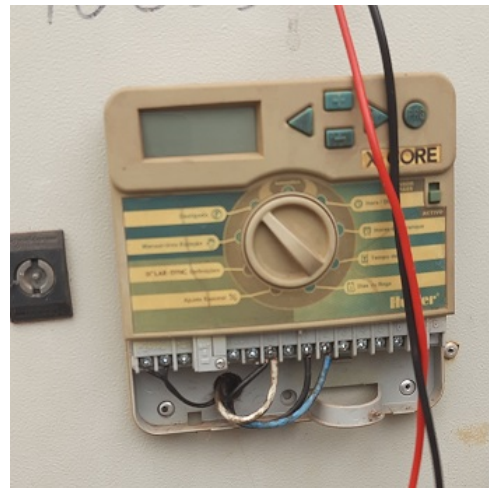
Com o tempo, optou-se por não priorizar sua instalação, focando exclusivamente no desenvolvimento do protótipo do projeto. Assim, o X-Core foi reinstalado mas não foi colocado em funcionamento. No entanto, testes preliminares foram realizados para verificar o funcionamento básico do equipamento.

A Figura 18 apresenta os painéis elétricos localizados na sala de controle do viveiro, onde o X-Core foi reinstalado. À esquerda (Figura 18(a)), observa-se o painel elétrico responsável pelo controle dos dispositivos do viveiro, a partir do qual foram elaborados os diagramas apresentados na Seção anterior. À direita encontra-se o painel do X-Core (Figura 18(b)).

Figura 18 – Painéis elétricos da sala de controle em que o X-Core estava instalado.



(a) – Painéis elétricos.



(b) – X-Core.

Fonte: Elaborado pela autora, 2024.

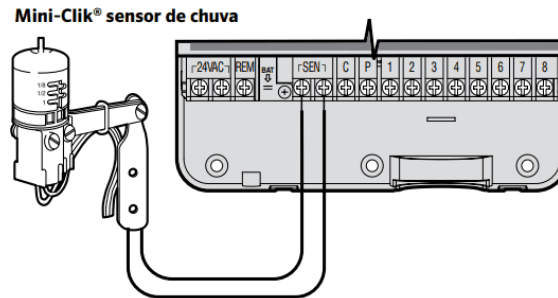
Embora o X-Core não tenha sido utilizado diretamente no projeto, sua análise foi fundamental para definir os requisitos do protótipo e identificar limitações a serem superadas.

Inicialmente, verificou-se que o X-Core não oferece conectividade *online*, sendo suas configurações feitas manualmente por meio de botões de pressão e um seletor, o que demanda a consulta frequente ao manual para ajustes. Para superar essa limitação, optou-se por desenvolver uma interface *web*, proporcionando um controle acessível e intuitivo dos equipamentos via dispositivo móvel.

Adicionalmente, foi observado que o X-Core permite programar a rega apenas em intervalos de 15 minutos (por exemplo: 09:00, 09:15, 09:30, 09:45). O sistema proposto neste trabalho oferece maior flexibilidade, permitindo que o usuário defina horários específicos para a irrigação, adaptando-a conforme necessário.

Outro aspecto observado foi a limitação no uso de sensores. O X-Core dispõe de apenas uma entrada de sensor, geralmente conectada a um sensor meteorológico fabricado pela Hunter ou similar, que suspende a irrigação automaticamente com base nas condições climáticas. A conexão desse sensor pode ser visualizada na Figura 19.

Figura 19 – Conexão do sensor no x-core.



Fonte: Adaptado do manual do usuário.

Outro fator importante que foi analisado é a respeito da funcionalidade de setorização presente no X-Core. Apesar de o equipamento permitir a programação de diferentes setores de irrigação, cada um com horários distintos, essa capacidade é limitada a quatro áreas. Em contraste, o protótipo desenvolvido possui uma estrutura expansível, permitindo a adição de vários setores adicionais, dependendo apenas da conexão a novos relés. Essa flexibilidade facilita a ampliação do sistema, caso seja necessário para a instituição, permitindo o controle de novos setores além dos já programados.

Além disso, observou-se que tanto o X-Core quanto a bomba de água são diretamente conectados à rede elétrica principal, operando a 220 volts. A válvula, no entanto, requer apenas 24 volts para funcionar. O X-Core já conta com um conversor de tensão embutido para fornecer a energia necessária à válvula.

Por outro lado, o ESP32, utilizado no protótipo, opera com uma tensão de 5 volts. Para garantir o funcionamento adequado do sistema, foi necessária a instalação de um regulador de tensão, capaz de reduzir os 220V da rede elétrica para 5V. O relé utilizado no projeto é responsável por ativar a válvula de 24V, sendo a tensão fornecida por um transformador que converte os 220V da rede elétrica em 24V.

5 RESULTADOS

Neste capítulo, serão apresentados os principais aspectos do desenvolvimento do projeto. Primeiramente, a Seção 5.1 descreve o processo de construção e montagem do sistema, abordando o desenvolvimento de cada componente e a estrutura física utilizada. Em seguida, na Seção 5.2, é detalhado o código, com explicações sobre as funcionalidades programadas e como elas permitem o funcionamento automatizado da irrigação. Por fim, na Seção 5.3, são descritas as etapas de instalação e integração do sistema, seguidas de uma análise dos resultados finais obtidos com o projeto completo.

5.1 Protótipo

Esta seção descreve o desenvolvimento e a construção física do protótipo do sistema. Inicia-se com a concepção do projeto, abordada na Subseção 5.1.1, onde são discutidas as diretrizes principais e a estruturação inicial das ideias.

A Subseção 5.1.2 apresenta a montagem preliminar no *proto-board*, etapa essencial para validar os componentes e conexões. Em seguida, a Subseção 5.1.3 aborda a utilização da plataforma Arduino IDE, detalhando a programação do microcontrolador ESP32 e sua integração com sensores e relés.

Na Subseção 5.1.4, é discutido o projeto da placa de circuito impresso utilizando o *software* EasyEDA. Por fim, a Subseção 5.1.5 explora o processo de montagem da placa definitiva, dividido em etapas para garantir clareza e compreensão ao leitor.

5.1.1 Concepção inicial do projeto

O desenvolvimento inicial do protótipo foi realizado utilizando a plataforma Wokwi, um simulador de eletrônica *online*. O Wokwi permite a simulação de diversos microcontroladores, como Arduino e ESP32, além de uma variedade de componentes, incluindo sensores, LEDs e relés.

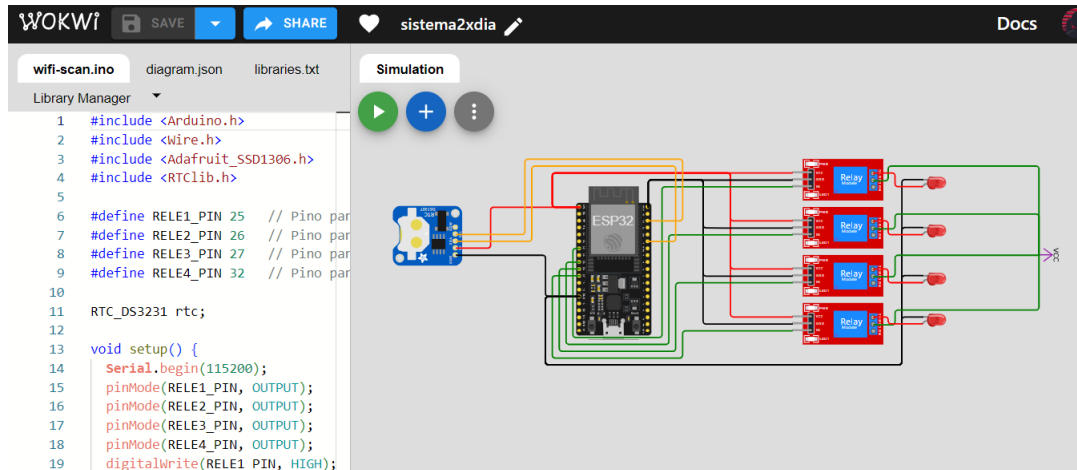
A escolha dessa plataforma se deu pela facilidade de acesso e pelo suporte a componentes como o ESP32, o módulo relé e o RTC. Inicialmente, o Wokwi foi utilizado para desenvolver o código responsável pelo controle do sistema com base nos horários programados. No entanto, os recursos disponíveis mostraram-se limitados, especialmente pela ausência de simulação dos sensores de umidade e temperatura, fundamentais para o projeto.

Apesar dessas restrições, o uso do Wokwi foi fundamental para o aprendizado e a familiarização com a programação do ESP32. A plataforma oferece tutoriais e exemplos de código, facilitando a implementação das primeiras funcionalidades do protótipo.

A Figura 20 demonstra a utilização da plataforma na versão inicial do código. À esquerda, observa-se o código em C++, que verifica o horário utilizando o RTC e aciona

o LED conectado ao módulo relé quando coincide com o horário programado. Esse código será detalhado na Subseção 5.2.1. À direita, podem ser visualizadas a simulação dos componentes, as conexões elétricas e os LEDs utilizados para representar o funcionamento do sistema.

Figura 20 – Uso do Wokwi para a primeira versão do protótipo e código.



Fonte: Elaborado pela autora, 2024.

5.1.2 Montagem no protoboard

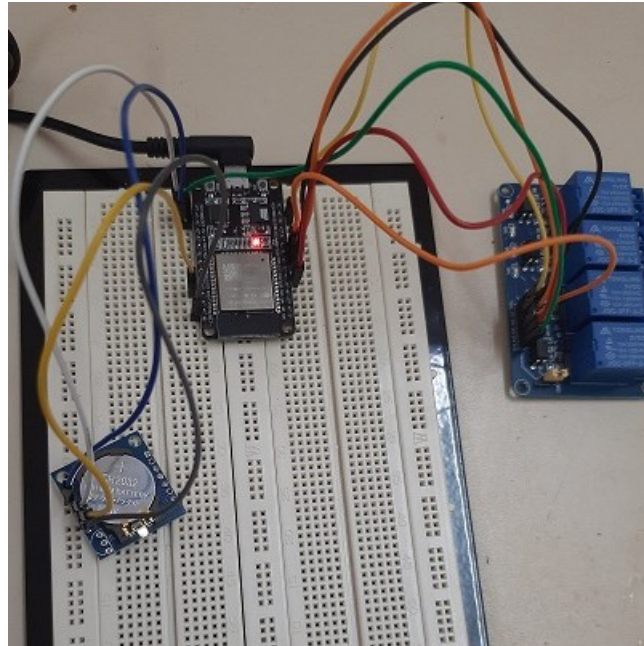
Após finalizar a versão do código que utilizava o Wokwi, iniciou-se a fase de montagem física do sistema, conectando os dispositivos para testes. Para essa etapa, utilizou-se uma placa de *protoboard* genérica. O ESP32 foi posicionado no centro da *protoboard*, e as conexões foram realizadas com o auxílio de fios *jumper*, ligando o módulo relé e o RTC ao microcontrolador. Os pinos necessários para as conexões já haviam sido definidos previamente durante a simulação no Wokwi, o que facilitou a montagem.

Durante o desenvolvimento do código no simulador, surgiram pequenos erros de lógica e nas pinagens. No entanto, a identificação precoce desses erros no ambiente virtual permitiu que fossem resolvidos antes da fase prática, garantindo que o código estivesse funcional na montagem física. Assim, ao realizar a montagem na *protoboard*, os problemas previamente observados na simulação não se repetiram.

As conexões realizadas podem ser observadas na Figura 21, que apresenta o protótipo montado na *protoboard*.

Com a montagem concluída, foi possível testar o sistema, confirmando que ele funcionava conforme o esperado.

Figura 21 – ESP32 conectado aos equipamentos no *protoboard*.



Fonte: Elaborado pela autora, 2024.

5.1.3 *Arduino IDE*

Após a análise da primeira etapa do desenvolvimento, identificou-se a necessidade de utilizar uma plataforma mais abrangente, optando pelo uso do Arduino IDE. Embora essa plataforma não permitisse a simulação do código de forma *online*, como o Wokwi, essa limitação não foi um obstáculo, uma vez que o protótipo físico já estava disponível para testes.

O código testado no Wokwi foi transferido para o Arduino IDE e, após confirmar que a versão básica estava funcionando corretamente, foram adicionados os sensores de temperatura do ar (DHT11) e umidade do solo (YL-69).

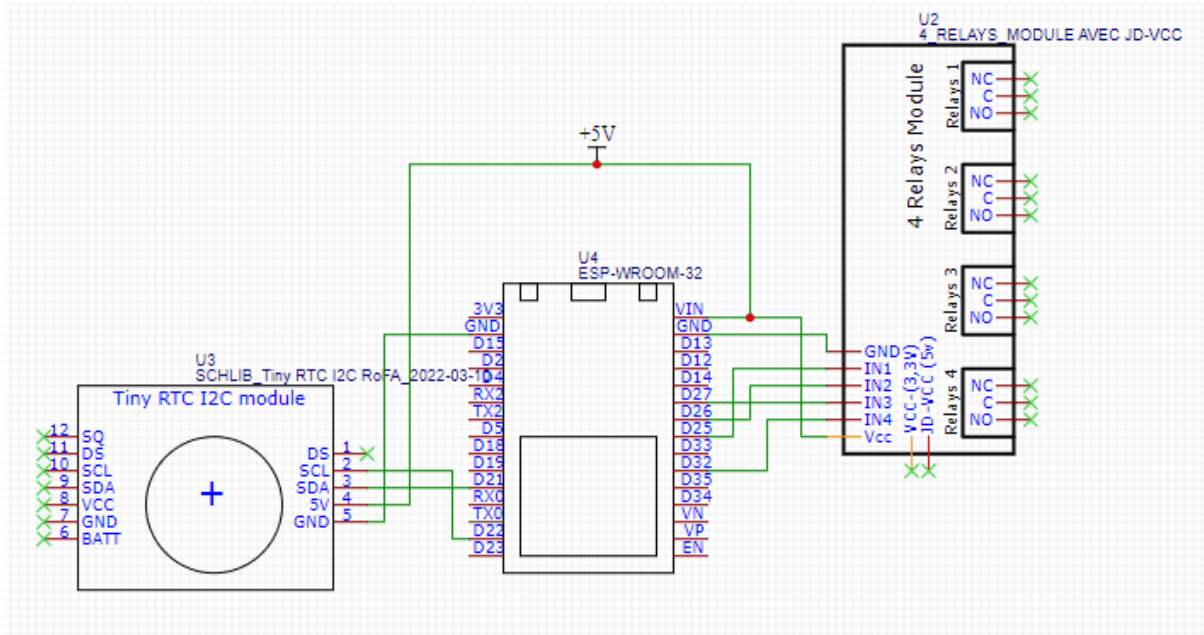
A nova versão do código, que poderá ser visto na Subseção 5.2.2, foi implementada no Arduino IDE, onde era possível programar um horário específico para o acionamento dos relés. Simultaneamente, o sistema realizava leituras dos sensores e exibia os valores em tempo real, proporcionando controle e monitoramento do ambiente.

Além disso, foi utilizada essa IDE para programar a versão final do código, que permitiu ao usuário controlar os relés e monitorar os dados dos sensores utilizando um dispositivo móvel. Essa etapa finalizou o desenvolvimento do código, e será explicado de forma mais detalhada na Seção 5.2.3.

5.1.4 Software Easyeda para projeto da placa

Antes de desenvolver a placa fisicamente, foi necessário utilizar um *software* para planejar e elaborar o circuito esquemático de forma mais completa. Para isso, foi utilizado o *software* EasyEDA. A Figura 22 apresenta o diagrama esquemático do circuito, que inclui um microcontrolador ESP32, um módulo RTC e um módulo de relés.

Figura 22 – Circuito esquemático do projeto elaborado no Easyeda.



Fonte: Elaborado pela autora, 2024.

As conexões foram feitas da seguinte forma: o pino "SCL" (*clock* do barramento I2C) foi conectado ao pino "D22" do ESP32, e o pino "SDA" (dados do barramento I2C) foi conectado ao pino "D21" do ESP32. Os pinos "VCC" e "GND" do módulo RTC foram conectados aos pinos correspondentes do ESP32 para fornecer a alimentação.

O módulo de relés (U2) é composto por quatro relés, que podem ser usados para controlar mais de um dispositivo ao mesmo tempo. Cada relé possui três terminais: "NO" (*Normally Open*), que estabelece a conexão quando o relé é acionado; "NC" (*Normally Closed*), que interrompe a conexão quando o relé é acionado; e "COM" (*Common*), que alterna entre "NO" e "NC".

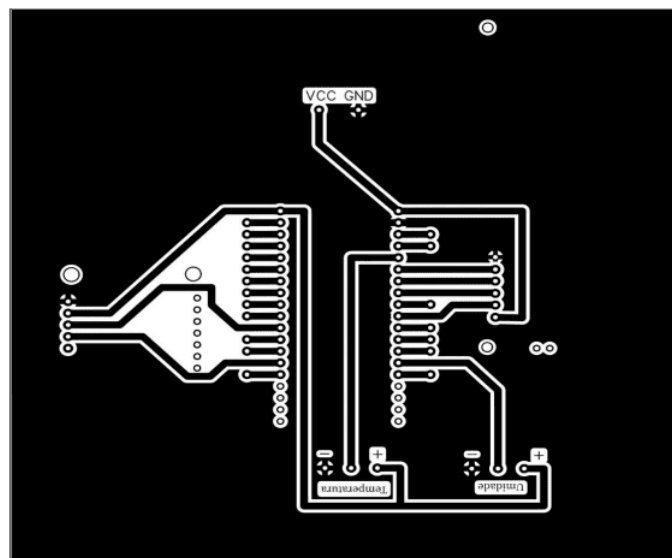
As conexões específicas do módulo de relés foram feitas da seguinte maneira: os pinos "IN1", "IN2", "IN3" e "IN4" do módulo de relés foram conectados aos pinos "D26", "D25", "D33" e "D32" do ESP32, respectivamente. Esses pinos controlam individualmente cada relé.

O circuito é alimentado por uma fonte de 5V, que distribui energia para o ESP32, o módulo RTC e o módulo de relés, garantindo o funcionamento contínuo do sistema.

Como os sensores não foram soldados diretamente na placa, mas sim conectados por meio de cabos, foi necessário reservar os pinos apropriados para eles. Os pinos "A0", "VCC" e "GND" do sensor de umidade do solo (YL-69) foram conectados aos pinos "A0", "3,3V" e "GND" do ESP32, respectivamente. Da mesma forma, o sensor de temperatura e umidade do ar (DHT11) teve seus pinos "+", "-", "OUT" conectados aos pinos "VCC", "GND" e "D14" do ESP32, respectivamente.

A Figura 23, é a representação de como o circuito ficou fisicamente montado na placa. Ela mostra as trilhas de cobre que conectam os componentes, a posição dos furos onde os componentes serão soldados e a organização geral dos componentes na placa.

Figura 23 – Circuito impresso desenvolvido no Easyeda para impressão na placa.



Fonte: Elaborado pela autora, 2024.

Após a geração da imagem da PCB, a placa foi confeccionada de acordo com os passos descritos na Subseção 5.1.5.

5.1.5 Montagem da placa

A seguir, será apresentado o processo de desenvolvimento e montagem da placa do protótipo, detalhando passo a passo as etapas envolvidas, desde impressão do circuito impresso até a finalização do protótipo físico.

1. Impressão da imagem:

A imagem do circuito impresso gerada anteriormente foi impressa em papel fotográfico. É importante garantir que a impressão seja espelhada em relação ao *layout* visualizado na tela no momento da impressão.

2. Transferência da imagem para a placa:

Para transferir o desenho da PCB para a placa de fenolite cobreada, foi utilizado um ferro de passar roupa para aquecer o papel fotográfico, garantindo que a imagem impressa fosse transferida para a superfície de cobre da placa (Figura 24). O lado a ser aplicado na placa deve ser o inverso do lado em que os componentes serão montados. Antes de iniciar a transferência, é essencial limpar a placa com lã de aço seca para remover sujeiras e garantir uma aderência adequada.

Figura 24 – Transferência do desenho no papel para a placa de circuito impresso.



Fonte: Elaborado pela autora, 2024.

3. Remoção do excesso de papel:

Após confirmar que o desenho foi corretamente transferido para a placa, o próximo passo consiste em remover o excesso de papel utilizando água. Durante esse processo, é fundamental expor apenas as áreas de cobre que serão corroídas. Para garantir uma remoção precisa, foi utilizado um objeto pontiagudo, evitando a remoção indesejada da parte impressa nas áreas mais delicadas. A Figura 25 ilustra a placa depois de retirar todo o excesso de papel e pronta para corroer.

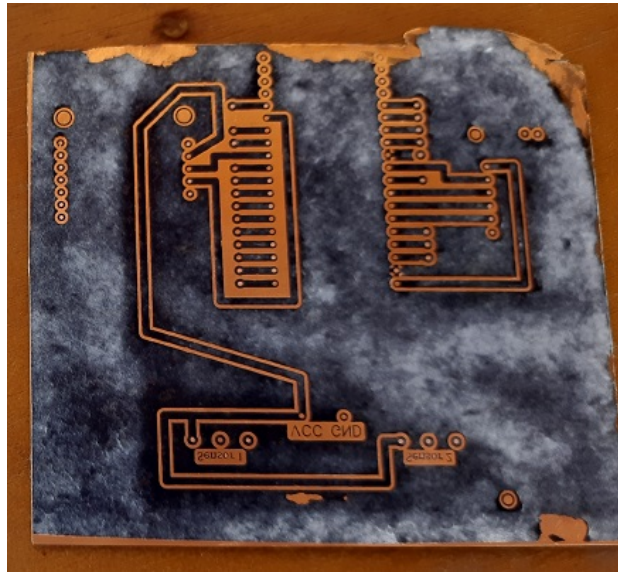
4. Corrosão da placa:

Para este processo é preciso de mergulhar a placa completamente no perclorato de ferro, um sal que, em solução aquosa, é eficaz na corrosão de placas de circuito impresso. A Figura 26 ilustra a placa completamente imersa na solução de perclorato de ferro, demonstrando o processo de corrosão em andamento.

5. Limpeza e correção das ligações:

Após o processo de corrosão, foi realizada a limpeza da placa para remover o papel utilizado na proteção do desenho do circuito elétrico. Essa remoção assegura que a área do circuito fique exposta para as próximas etapas de montagem.

Figura 25 – Placa de fenolite após a transferência do desenho e a limpeza do excesso de papel.



Fonte: Elaborado pela autora, 2024.

Figura 26 – Corrosão da placa no percloroeto de ferro.



Fonte: Elaborado pela autora, 2024.

6. Perfuração da placa:

Para criar os furos na placa, foi utilizado um perfurador de placa de circuito impresso. Essa ferramenta facilitou a realização dos furos necessários para a inserção dos componentes eletrônicos. É importante destacar que o projeto foi programado pensando em possíveis expansões futuras, por isso, foram feitos furos adicionais.

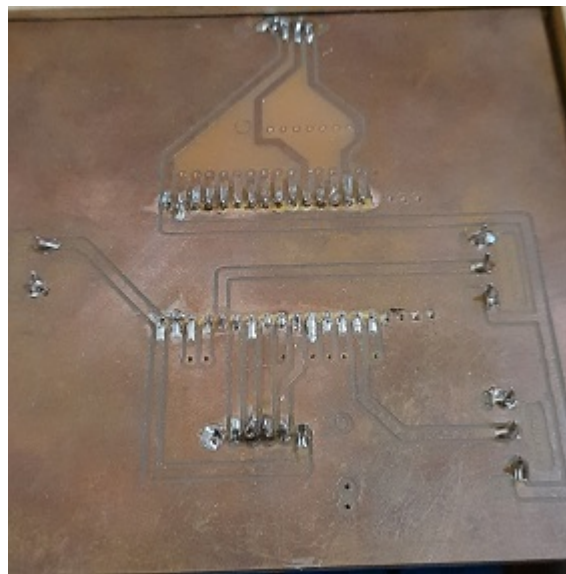
7. Encaixe dos equipamentos:

Após a perfuração da placa, os equipamentos foram encaixados conforme o planejamento. As ligações dos *jumpers* foram realizadas diretamente na placa, a fim de facilitar a conexão dos sensores posteriormente.

8. Soldagem dos fios e equipamentos:

Em seguida, os equipamentos, *jumpers* e a fonte de alimentação foram soldados na placa. Para esse processo, foi utilizado um ferro de solda, estanho e equipamentos de suporte, todos disponíveis no laboratório. A Figura 27 retrata a placa após a soldagem.

Figura 27 – Placa de fenolite após a soldagem dos componentes.



Fonte: Elaborado pela autora, 2024.

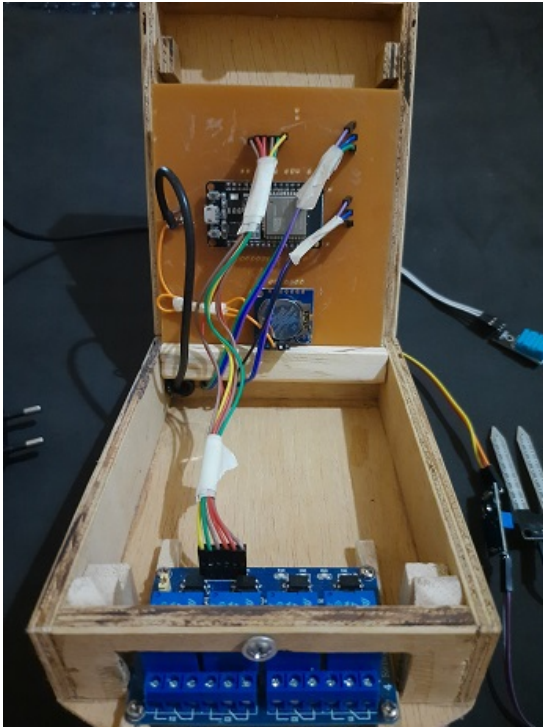
9. Confecção da caixinha:

Como último passo, foi produzida uma caixinha de madeira para acomodar a placa. O *design* foi elaborado para deixar exposta apenas a parte de conexão dos fios do relé, enquanto os fios de energia e dos sensores ficam protegidos na parte interna.

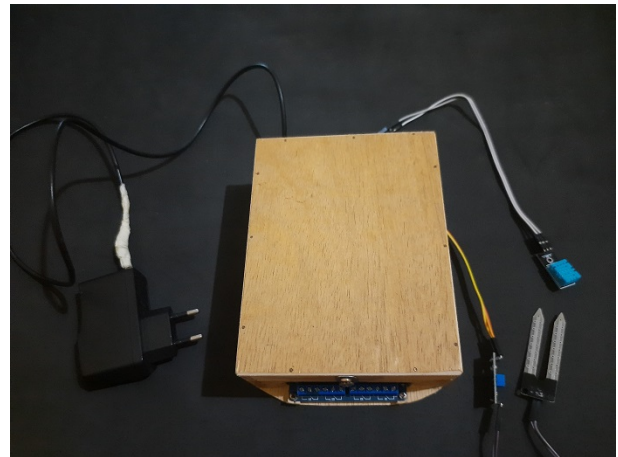
10. Protótipo finalizado:

A Figura 28 representa o protótipo finalizado, pronto para a instalação. Na Figura 28(a) é possível observar o protótipo com a tampa aberta, facilitando o acesso aos componentes para possíveis ajustes. Já na Figura 28(b) é mostrado o protótipo fechado, indicando como ficará após a instalação, permitindo acesso apenas ao relé, à fonte e aos sensores, que são as partes necessárias para manuseio durante a instalação.

Figura 28 – Protótipo finalizado.



(a) Protótipo com a tampa aberta.



(b) Protótipo com a tampa fechada.

Fonte: Elaborado pela autora, 2024.

5.2 Código

Nesta seção, os códigos desenvolvidos ao longo do projeto são apresentados. A Seção 5.2.1 foi implementado na plataforma Wokwi, conforme descrito anteriormente.

O segundo código, Seção 5.2.2, foi desenvolvido com base no primeiro, sendo reestruturado para incluir as funções de leitura dos sensores.

O código principal, Seção 5.2.3, atualmente em execução no microcontrolador, também foi elaborado a partir do segundo código, incorporando funcionalidades para a conexão com a internet.

Essa abordagem de dividir o desenvolvimento em diferentes versões de código foi fundamental para o estudo das tecnologias empregadas, facilitando o progresso do projeto. Essa estratégia garantiu que cada funcionalidade fosse implementada e testada de forma isolada, permitindo a identificação e resolução de eventuais problemas antes de avançar para a versão final.

Os detalhes dos códigos poderão ser vistos nas subseções abaixo.

5.2.1 Controle do relé

O código apresentado tem o objetivo de controlar o acionamento do relé de acordo com horários previamente programados. Ele monitora o horário atual e ativa ou desativa o relé conforme os intervalos de tempo definidos.

Nas primeiras linhas, foram incluídas três bibliotecas que fornecem as funções necessárias para controlar o *hardware* e realizar a comunicação entre o microcontrolador e os módulos externos. Em seguida, foi realizada a definição do pino do relé (CÓDIGO 1).

Código 1 – Bibliotecas e definições dos pinos - primeiro código.

```
#include <Arduino.h>
#include <Wire.h>
#include <RTClib.h>
#define RELE1_PIN 25
RTC_DS3231 rtc;
```

Fonte: Elaborado pela autora, 2024.

Na função *setup* (Código 2), são configuradas as funcionalidades do sistema, incluindo a definição do pino conectado ao relé, garantindo que o relé seja iniciado em estado desligado. Além disso, o código verifica o módulo RTC, assegurando que ele esteja operando corretamente.

Código 2 – Função *setup* - primeiro código.

```
void setup() {
  Serial.begin(115200);
  pinMode(RELE1_PIN, OUTPUT);
  digitalWrite(RELE1_PIN, HIGH);
  if (!rtc.begin()) {
    Serial.println("Erro ao iniciar o RTC!");
    while (1);
  }
  if (rtc.lostPower()) {
    Serial.println("RTC perdeu a hora, reajustando...");
  }
}
```

Fonte: Elaborado pela autora, 2024.

Na função *loop* (Código 3), o código executa um ciclo contínuo em que obtém a hora atual através do módulo RTC, armazena os valores de hora e minuto e exibe esses dados no monitor serial.

Código 3 – Função *loop* - primeiro código.

```

void loop() {
    DateTime now = rtc.now();
    int hour = now.hour();
    int minute = now.minute();
    Serial.print(hour);
    Serial.println(":");
    Serial.println(minute);
    // Define os horários de acionamento do relé
    int rele1_hour = 12;    // Hora para acionar o relé
    int rele1_minute = 10; // Minuto para acionar o relé
    // Aciona o relé de acordo com o horário
    if (hour == rele1_hour && minute == rele1_minute) {
        digitalWrite(RELE1_PIN, LOW);
    } else {
        digitalWrite(RELE1_PIN, HIGH);
    }
    delay(1000);
}

```

Fonte: Elaborado pela autora, 2024.

Com base no horário atual, o código compara esses valores com o horário programado para ativar o relé. Se a hora e o minuto atuais coincidirem com o horário predefinido, o relé é ativado, acionando o dispositivo externo conectado. Caso contrário, o relé permanece desligado. Por fim, foi adicionado um *delay* de 1 segundo, para evita que o código faça verificações muito rápidas e excessivas, além de permitir que o horário seja verificado apenas uma vez por segundo.

Esse código serviu como base para o desenvolvimento subsequente, no qual foram adicionadas novas funcionalidades, como a integração de sensores e a conexão via *web*. A evolução desse código, com a implementação do monitoramento de sensores, será abordada na Subseção 5.2.2.

5.2.2 Monitoramento dos sensores

Os códigos apresentados nesta subseção foi desenvolvido com base nos códigos anteriores, com a adição de bibliotecas e funções que aprimoram o controle do sistema.

O Código 4 inicia com a inclusão de novas bibliotecas e as definições dos pinos para o sensor de temperatura e umidade *DHT* e para o sensor de umidade do solo.

Além disso, foram criadas variáveis globais para guardar leituras de estados que serão acessados em várias partes do código, evitando a necessidade de passar como parâmetro em cada função. Elas podem ser vistas no Código 5.

Código 4 – Definição das bibliotecas e dos pinos - segundo código.

```
#include <RTClib.h>
#include <DHT.h>
#define RELE1_PIN 27
#define SOIL_HUMIDITY_PIN A0
#define DHT_PIN 14
#define DHT_TYPE DHT11
```

Fonte: Elaborado pela autora, 2024.

Elas tem o objetivo de:

- Armazenar a hora atual e da última irrigação;;
- Registrar os valores lidos pelos sensores;
- Armazenar valores para calibração da irrigação;
- Indicar o momento da próxima irrigação;
- Verificar se a irrigação está ativa;
- Controlar o tempo de irrigação - registro do início e duração da irrigação.

Código 5 – Definição das variáveis - Segundo Código.

```
DHT dht(DHT_PIN, DHT_TYPE);
RTC_DS1307 rtc;
int current_hour; // hora atual
int current_minute; // minuto atual
int aux = 0;
float soil_humidity = 0;
float air_temperature = 0;
float air_humidity = 0;
int leituraSeca = 0;
int leituraUmida = 0;
int last_hour = 0; // variável pra guardar a última hora
int last_minute = 0;
int next_hour = 15;
int next_minute = 46;
bool regaAtiva = false;
unsigned long tempoInicioRega;
unsigned long duracaoRega = 60000; //em milissegundos
```

Fonte: Elaborado pela autora, 2024.

Para organizar melhor o código, foram criadas funções específicas. O Código 6 implementa uma função responsável por controlar o acionamento e o desligamento do sistema de irrigação com base em um horário programado e uma duração específica para a irrigação.

Código 6 – Definição da função para ligar a rega automática - segundo código.

```
void LigarRega() {
    if (current_hour == next_hour &&
        current_minute == next_minute && !regaAtiva) {
        Serial.println("Rega ligada");
        digitalWrite(RELE1_PIN, LOW);
        tempoInicioRega = millis();
        regaAtiva = true;
    }
    // Verifica se a rega deve ser desligada
    if (regaAtiva && millis() - tempoInicioRega >= duracaoRega) {
        Serial.println("Rega desligada");
        digitalWrite(RELE1_PIN, HIGH);
        regaAtiva = false;
    }
}
```

Fonte: Elaborado pela autora, 2024.

A função verifica se a hora e o minuto atuais coincidem com o horário definido para o início da irrigação. Além disso, checka se a irrigação está desligada, evitando que seja acionada novamente se já estiver em funcionamento. Caso as condições sejam atendidas, o sistema ativa o relé. Após a ativação, a função monitora o tempo decorrido e, se este ultrapassar a duração estipulada, desliga automaticamente a irrigação.

Também foram implementadas funções para o controle manual da irrigação, permitindo ao usuário ativar e desativar a rega diretamente. A primeira função (Código 7) aciona o relé responsável por abrir a válvula de água, iniciando a irrigação. Já a segunda função (Código 8) desativa o relé, interrompendo o fluxo de água e desligando a irrigação.

Código 7 – Definição da função para ligar a rega manual - segundo código.

```
void ligarRegaManual() {
    Serial.println("Rega manual ligada");
    digitalWrite(RELE1_PIN, LOW);
}
```

Fonte: Elaborado pela autora, 2024.

Código 8 – Definição da função para desligar a rega manual - segundo código.

```
void desligarRegaManual() {
    Serial.println("Rega manual desligada");
    digitalWrite(RELE1_PIN, HIGH);
}
```

Fonte: Elaborado pela autora, 2024.

Para realizar a calibração do sensor de umidade do solo, foi criada a função apresentada no Código 9. Ela assegura que a calibração ocorra apenas uma vez, evitando múltiplas calibrações desnecessárias a cada execução do código, o que poderia redefinir os valores de referência sem necessidade. Durante a execução, a função orienta o usuário a inserir o sensor de umidade em solo seco. Após um atraso de dez segundos, a leitura é realizada e armazenada como o valor de referência para o solo seco. O mesmo procedimento é repetido para o solo úmido, definindo o valor de referência correspondente.

Código 9 – Função para calibrar o sensor de umidade do solo - segundo código.

```
void calibrarSensorUmidadeSolo() {
    if (aux == 0){
        Serial.println("Insira o sensor em solo seco e aguarde...");
        delay(10000);
        leituraSeca = analogRead(SOIL_HUMIDITY_PIN);
        Serial.println("Leitura seca concluída");
        delay(1000);
        Serial.println("Insira o sensor em solo úmido e aguarde...");
        delay(10000);
        leituraUmida = analogRead(SOIL_HUMIDITY_PIN);
        Serial.println("Leitura umida concluída");
        delay(1000);
        Serial.println("Calibração concluída.");
        aux = aux + 1;
    }
}
```

Fonte: Elaborado pela autora, 2024.

O sensor de umidade do solo (YL-69) requer uma calibração inicial porque mede a umidade de forma indireta, detectando a resistência elétrica entre seus terminais. Essa resistência pode variar dependendo do tipo de solo, da compactação e de outras condições ambientais. Já o sensor DHT11, que mede a temperatura e a umidade do ar, é pré-calibrado e fornece valores absolutos diretamente.

A função apresentada no Código 10 tem o objetivo de atualizar os valores dos

sensores de temperatura, umidade do ar e umidade do solo. Ela inicia lendo os valores de umidade e temperatura utilizando o sensor DHT11. Essas leituras são armazenadas nas variáveis *Air Humidity* e *Air Temperature*, respectivamente. Em seguida, o sistema realiza a leitura do valor analógico do sensor de umidade do solo (YL-69). A função também obtém o horário atual do RTC, atualizando as variáveis *Current Hour* e *Current Minute*.

Código 10 – Parte da função (*updateSensorsValue*) que recebe as leituras dos sensores para atualizar os valores dos sensores - segundo código.

```
void updateSensorsValue() {
    air_humidity = dht.readHumidity();
    air_temperature = dht.readTemperature();
    soil_humidity = analogRead(SOIL_HUMIDITY_PIN);
    if (isnan(air_humidity) || isnan(air_temperature)) {
        Serial.println("Falha na leitura do DHT!");
        return;
    }
    DateTime now = rtc.now();
    current_hour = now.hour();
    current_minute = now.minute();
}
```

Fonte: Elaborado pela autora, 2024.

Por fim, a função converte a leitura do sensor de umidade do solo em um valor percentual (Código 11). Para isso, utiliza-se a função *map* para ajustar os valores do sensor. Os limites de calibração da leitura seca e leitura úmida, são mapeados para uma escala de zero a cem, transformando a leitura do sensor em uma porcentagem de umidade do solo, onde 0% indica solo seco e 100% representa solo completamente úmido.

Código 11 – Parte da função (*updateSensorsValue*) que converte os valores - segundo código.

```
DateTime now = rtc.now();
current_hour = now.hour();
current_minute = now.minute();
soil_humidity = map(soil_humidity, leituraSeca, leituraUmida, 0, 100);
Serial.print("Temp: ");
Serial.print(air_temperature);
Serial.print(" Hum: ");
Serial.print(air_humidity);
Serial.print(" Soil: ");
Serial.println(soil_humidity);
```

Fonte: Elaborado pela autora, 2024.

O código 12 é responsável pela inicialização do *hardware* e dos sensores utilizados no sistema. Ele configura o comportamento dos pinos e do relé, além de chamar a função de calibração do sensor de umidade do solo. A inicialização do RTC também ocorre nesta função, garantindo que o sistema esteja pronto para operar corretamente.

Código 12 – Configuração dos pinos e inicialização dos dispositivos na função *setup* - segundo código.

```
void setup() {
  Serial.begin(115200);
  pinMode(RELE1_PIN, OUTPUT);
  pinMode(SOIL_HUMIDITY_PIN, INPUT);
  digitalWrite(RELE1_PIN, HIGH);
  calibrarSensorUmidadeSolo();
  Serial.println("Iniciando o sensor DHT...");
  dht.begin(); // Inicializa o sensor DHT
  delay(2000);
  if (!rtc.begin()) {
    Serial.println("Erro ao iniciar o RTC!");
    while (1);
  }
}
```

Fonte: Elaborado pela autora, 2024.

Por fim, temos o *loop* principal do programa, apresentado no Código 13, onde as leituras dos sensores e o controle da irrigação são executados repetidamente. O programa também imprime algumas informações no monitor serial possibilitando o monitoramento e a depuração.

Código 13 – função *loop* - segundo código.

```
void loop() {
  updateSensorsValue();
  LigarRega();
  Serial.println("proxima hora");
  Serial.println(next_hour);
  Serial.println("proximo minuto");
  Serial.println(next_minute);
  Serial.println("rega ativa");
  Serial.println(regaAtiva);
  delay(1000);
}
```

Fonte: Elaborado pela autora, 2024.

Em resumo, o código apresentado nesta seção constitui um sistema automatizado de irrigação que utiliza um microcontrolador para monitorar as condições ambientais e acionar a irrigação em horários específicos. O sistema permite também a calibração dos sensores e o controle manual da irrigação, com informações exibidas no monitor serial. Na Subseção 5.2.3, será apresentada uma versão aprimorada desse código, com melhorias nas funções existentes e a adição de novas funções para conexão *web*.

5.2.3 *Controle pela web*

O código descrito nesta subseção representa a versão final do protótipo. Seu objetivo principal é monitorar os sensores de umidade do solo, temperatura e umidade do ar, além de permitir o controle automático e manual da irrigação.

A frequência de irrigações diárias não é fixa, permitindo flexibilidade ao sistema. O sistema aceita até dois ciclos de irrigação por vez e, após a execução de cada ciclo, tem a possibilidade de ajustar as configurações e adicionar mais irrigações no mesmo dia, se necessário. O último horário configurado permanecerá salvo até ser atualizado, evitando a necessidade de reprogramações diárias. Atualmente o sistema aceita apenas dois ciclos por dia, mas essa quantidade pode ser alterada por meio de reprogramação, caso seja desejado no futuro.

Para essa versão do protótipo e durante os testes foram pré-definidos dois horários, um na parte da manhã e outro na parte da tarde. Essa configuração garante que, em caso de reinicialização do sistema, a irrigação não seja interrompida até que o usuário possa intervir.

O projeto já tem a possibilidade de expansão. Desta forma, para trabalhos futuros, o protótipo poderá controlar múltiplos relés. No projeto atual, conforme já explicitado, tem-se o controle de dois atuadores - a bomba e da válvula do setor 3. Logo, o código foi feito para acionar dois relés. O código completo está no Apêndice A.

A inclusão das bibliotecas e definições dos pinos é semelhante às aquelas apresentadas no Código 4, mas com a adição das bibliotecas *Wi-Fi* e *WebServer*, que são responsáveis pela conexão à rede e pela criação de um servidor. Além disso, nas definições de pinos foi adicionado mais um pino para controlar o segundo relé, permitindo uma expansão das funcionalidades do sistema.

Foi incluída a definição do identificador de conjunto de serviços - *SSID* (do inglês *Service Set Identifier*), que corresponde ao nome da rede, juntamente com a senha para conexão ao roteador. Além disso, foi configurado um IP fixo, bem como o *gateway* e a máscara de sub-rede, garantindo a comunicação adequada do sistema na rede. Essa configuração pode ser visualizada no Código 14.

As variáveis globais são semelhantes às do Código 5, mantendo funcionalidades como a ativação automática baseada em horários programados, leituras dos sensores e a

Código 14 – Definição da conexão com a internet - terceiro código.

```
const char* ssid = "projeto-viveiro";
const char* password = "88JA4uEjYL";
IPAddress ip(10, 60, 63, 250);
IPAddress gateway(10, 60, 60, 1);
IPAddress subnet(255, 255, 252, 0);
WebServer server(80);
```

Fonte: Elaborado pela autora, 2024.

calibração do sensor de umidade do solo. A novidade agora é a inclusão de um segundo horário programável para a irrigação.

Além disso, foi adicionada uma função que controla o acionamento e desligamento automáticos do sistema de irrigação com base no horário atual e na duração programada (Código 15). O sistema verifica se o horário atual coincide com um dos horários agendados para o início da irrigação. Quando essa condição é atendida, os relés são ativados, iniciando a irrigação. O horário de início é registrado e a variável que indica a irrigação ativa é ajustada para "verdadeira", sinalizando que o processo está em execução. O horário da última irrigação é então atualizado de acordo com o horário atual.

Código 15 – função para controlar a rega de acordo com as horas - terceiro código.

```
void LigarRega() {
if ((current_hour == next_hour && current_minute == next_minute ||
    current_hour == next_hour2 && current_minute ==
    next_minute2) && !regaAtiva) {
    Serial.println("Rega ligada");
    digitalWrite(RELE1_PIN, LOW);
    delay(2000);
    digitalWrite(RELE2_PIN, LOW);
    tempoInicioRega = millis(); // Registra o tempo de início da rega
    regaAtiva = true;
    last_hour = current_hour;
    last_minute = current_minute;
}
if (regaAtiva && millis() - tempoInicioRega >= duracaoRega) {
    Serial.println("Rega desligada");
    digitalWrite(RELE2_PIN, HIGH);
    delay(2000);
    digitalWrite(RELE1_PIN, HIGH);
    regaAtiva = false;
}
}
```

Fonte: Elaborado pela autora, 2024.

A função descrita acima tem o mesmo propósito que a apresentada no Código 6, com a diferença de que agora foi incluída uma verificação para o segundo horário e o controle de um segundo relé. Ela também verifica se a irrigação está ativa e se a duração programada foi atingida. Caso o tempo estabelecido seja concluído, os relés são desativados, interrompendo a irrigação.

É importante ressaltar que, com a adição de um relé específico para a bomba, foi inserido um *delay* de dois segundos para evitar sobrecarga nos canos devido a um excesso de água sem saída adequada. Portanto, quando a função é chamada, ela verifica as condições e, caso seja necessário acionar os dispositivos, ativa primeiro a válvula, aguardando dois segundos antes de acionar a bomba. O mesmo processo ocorre ao desligar, primeiro desativando a bomba e, após dois segundos, desligando a válvula.

As funções de controle manual foram atualizadas em relação ao Código 7, mas ainda permitem que o usuário ative ou desative manualmente o sistema de irrigação. No Código 16, é possível observar a adição de um segundo relé e um *delay* entre o acionamento do primeiro e do segundo relé, garantindo que a válvula seja ativada antes da bomba. O mesmo procedimento é realizado para o desligamento, desativando a bomba antes da válvula.

Código 16 – Liga e Desliga a rega manualmente - terceiro código.

```
void ligarRegaManual() {
  Serial.println("Rega manual ligada");
  digitalWrite(RELE1_PIN, LOW);
  delay(2000);
  digitalWrite(RELE2_PIN, LOW);
  regaAtiva = true;
  tempoInicioRega = millis();
  last_hour = current_hour;
  last_minute = current_minute;
}

void desligarRegaManual() {
  Serial.println("Rega manual desligada");
  digitalWrite(RELE2_PIN, HIGH);
  delay(2000);
  digitalWrite(RELE1_PIN, HIGH);
  regaAtiva = false;
}
```

Fonte: Elaborado pela autora, 2024.

A função de calibração dos sensores de umidade do solo (Código 9) foi trazida do código anterior sem modificações, pois já cumpria seu objetivo de forma satisfatória, garantindo que a calibração ocorresse corretamente. A função responsável por atualizar os

valores dos sensores e exibir essas informações ao usuário também permaneceu inalterada, conforme apresentado no Código 10.

A principal adição neste código foi a função *Handle Root*, descrita no Código 17. Essa função utiliza HTML para gerar a interface da página com o usuário, exibindo as leituras dos sensores, horários de irrigação, o status atual da irrigação e controles para configurar a próxima rega ou ativar/desativar o sistema manualmente.

Código 17 – Função que gera a página em HTML para acesso remoto - terceiro código.

```
void handleRoot() {
    updateSensorsValue();
    // Inicia a criação da página HTML
    String page = "<html><head><title>Controle do
        Viveiro</title>";
    page += "<meta charset='UTF-8'>";
    page += "<style>";
    page += "body { font-family: Arial,
        sans-serif; margin: 0; padding: 0; }";
    page += ".container { padding: 20px; }";
    page += "h1 { background-color: #4CAF50;
        color: white; padding: 10px; }";
    page += ".section { border: 1px solid black;
        padding: 10px; margin-bottom: 20px; }";
    page += ".section h2 { margin-top: 0; }";
    page += "input[type='submit'] { background-color:
        #4CAF50; color: white; border: none; padding:
        0px 20px; cursor: pointer; border-radius: 12px;}";
    page += "input[type='submit']:hover { background-color: #45a049; }";
    page += "</style>";
    page += "</head><body><div class='container'>";
    page += "<h1>Controle    do Viveiro de Mudas</h1>";
```

Fonte: Elaborado pela autora, 2024.

Essa função é acionada sempre que o servidor recebe uma requisição HTTP para a página principal. Ela cria uma página HTML que exibe os dados dos sensores, horários de irrigação e o status atual do sistema, além de oferecer a possibilidade de controle manual e configuração dos horários. Antes de gerar a página, o sistema executa a função que atualiza os valores dos sensores, garantindo que as informações de temperatura, umidade do solo e umidade do ar estejam atualizadas.

O código também configura o cabeçalho da página HTML, definindo o título e a codificação UTF-8 para assegurar a compatibilidade com caracteres especiais. Adicionalmente, são incluídos estilos utilizando folha de estilo em cascatas - CSS (do inglês *Cascading Style Sheets*) com regras básicas de formatação, como fontes, margens,

espaçamentos e estilos para botões e contêineres, com o objetivo de tornar a interface mais amigável e visualmente atraente.

Em seguida, essa função exibe as informações dos sensores, conforme apresentado no Código 18. A interface mostra os dados coletados pelos sensores de temperatura do ar, umidade do ar e umidade do solo, todos conectados ao ESP32. Além disso, são exibidos a hora atual (obtida do módulo RTC), o horário da próxima irrigação programada e a última vez que a irrigação foi ativada.

Código 18 – Seção dos sensores e horários, da função *handle root* - terceiro código.

```
page += "<div class='section'>";
page += "<h2>Sensores</h2>";
page += "<p>Temperatura do ar: " + String(air_temperature) + " °C</p>";
page += "<p>Umidade do ar: " + String(air_humidity) + " %</p>";
page += "<p>Umidade do solo: " + String(soil_humidity) + " %</p>";
page += "</div>";

page += "<div class='section'>";
page += "<h2>Horários</h2>";
page += "<p>Hora atual: " + String(current_hour) + ":" +
        String(current_minute) + "</p>";
page += "<p>Horário das regas: " + String(next_hour) + ":" +
        String(next_minute) + " e " + String(next_hour2) + ":" +
        String(next_minute2) + "</p>";
page += "<p>Horário da última rega: " + String(last_hour) + ":" +
        String(last_minute) + "</p>";
page += "</div>";
```

Fonte: Elaborado pela autora, 2024.

Em seguida, a função exibe o estado atual do sistema de irrigação, indicando se a irrigação está ativa ou não, além de mostrar a duração programada para cada ciclo de irrigação. Essa seção é demonstrada no Código 19.

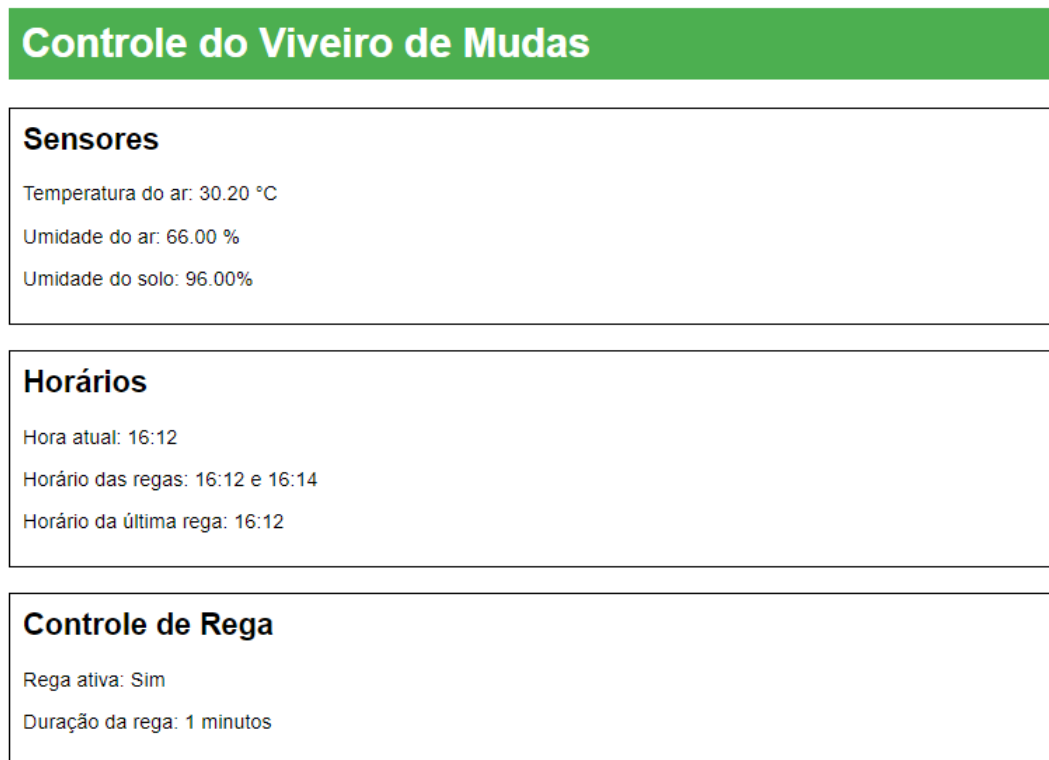
Código 19 – Seção de controle da rega, da função *handle root* - terceiro código.

```
page += "<div class='section'>";
page += "<h2>Controle de Rega</h2>";
page += "<p>Rega ativa: " + String(regaAtiva ? "Sim" : "Não") + "</p>";
page += "<p>Duração da rega: " + String(duracaoRega / 60000) + "
        minutos</p>";
page += "</div>";
```

Fonte: Elaborado pela autora, 2024.

Com o sistema em funcionamento, a interface *web* exibida na Figura 29 apresenta todas as funcionalidades mencionadas anteriormente. As seções referentes aos sensores, horários e controle de irrigação estão dispostas de forma clara e organizada, facilitando a navegação e a compreensão pelo usuário. A imagem foi capturada durante o uso real do sistema, destacando alguns aspectos relevantes. Primeiramente, os valores de temperatura do ar, umidade do ar e umidade do solo estavam sendo monitorados e exibidos em tempo real, conforme demonstrado na figura. Em segundo lugar, o "horário atual" coincidiu com o horário programado para irrigação, resultando na atualização automática do registro da última irrigação, refletindo o momento exato em que o sistema foi acionado. Por fim, como o horário atual permanecia dentro do período de irrigação definido, o status do sistema era exibido como ativo.

Figura 29 – Parte da página *web* com o monitoramento dos sensores e dos horários da rega.



Controle do Viveiro de Mudas

Sensores

Temperatura do ar: 30.20 °C

Umidade do ar: 66.00 %

Umidade do solo: 96.00%

Horários

Hora atual: 16:12

Horário das regas: 16:12 e 16:14

Horário da última rega: 16:12

Controle de Rega

Rega ativa: Sim

Duração da rega: 1 minutos

Fonte: Elaborado pela autora, 2024.

Posteriormente, foi criada uma área dedicada ao monitoramento e às configurações de irrigação. Nessa área, foi desenvolvido um formulário HTML que permite ao usuário atualizar o horário de irrigação, inserindo o próximo horário e a duração diretamente no formulário. Ao clicar no botão "Atualizar horário de rega", os dados são enviados ao servidor por meio de uma requisição *GET*. Essa funcionalidade pode ser visualizada no Código 20.

Código 20 – Seção de configurações do sistema - terceiro código.

```

page += "<div class='section'>";
page += "<h2>Configuração</h2>";
page += "<form action='/set_schedule' method='GET'>"; // Formulário
para definir o próximo horário de rega
page += "Primeiro horário de rega: <input type='number' name='hour1'
min='0' max='23' value='" + String(next_hour) + "'><br>";
page += "Primeiro minuto de rega: <input type='number' name='minute1'
min='0' max='59' value='" + String(next_minute) + "'><br>";
page += "Segundo horário de rega: <input type='number' name='hour2'
min='0' max='23' value='" + String(next_hour2) + "'><br>";
page += "Segundo minuto de rega: <input type='number' name='minute2'
min='0' max='59' value='" + String(next_minute2) + "'><br>";
page += "Tempo de rega (minutos): <input type='number' name='time'
min='1' value='" + String(duracaoRega / 60000) + "'><br>";
page += "<input type='submit' value='Atualizar horário de rega'>";
page += "</form>";
page += "</div>";

```

Fonte: Elaborado pela autora, 2024.

Além disso, foi criada uma seção com dois botões para o usuário: um para ativar e outro para desativar a irrigação manualmente. As ações correspondentes são enviadas ao servidor por meio de requisições *POST*. Esta é a última parte da função *handle root* e pode ser visualizada no Código 21.

Código 21 – Seção de controle manual - terceiro código.

```

// Seção de controle manual
page += "<div class='section'>";
page += "<h2>Rega Manual</h2>";
page += "<form action='/ligar_rega_manual' method='POST'>";
page += "<input type='submit' value='Ligar Rega Manual'>";
page += "</form>";
page += "<form action='/desligar_rega_manual' method='POST'>";
page += "<input type='submit' value='Desligar Rega Manual'>";
page += "</form>";
page += "</div>";
page += "</div></body></html>";
server.send(200, "text/html", page);
// Envia a página HTML para o cliente
}

```

Fonte: Elaborado pela autora, 2024.

A Figura 30 apresenta parte das configurações que podem ser feitas no site.

A Figura 30(a) mostra a seção gerada na página HTML, que é enviada ao usuário como uma página *web*. Nela, o usuário pode visualizar informações sobre a irrigação, configurar horários e acessar os botões para controle manual da rega.

Vale destacar que, durante o desenvolvimento do código, foram implementadas restrições para evitar erros por parte do usuário. Um exemplo disso pode ser visto na Figura 30(b), que ilustra o caso em que o usuário insere um período de tempo superior ao permitido (considerando os horários do dia). Nesse caso, uma mensagem de aviso é exibida e a requisição não pode ser enviada.

Figura 30 – Parte da página HTML gerada pela função *handle root*.

A interface é dividida em duas seções principais. A primeira, intitulada 'Configuração', contém quatro campos de entrada para configurar a rega: 'Primeiro horário de rega' (valor 17), 'Primeiro minuto de rega' (valor 23), 'Segundo horário de rega' (valor 40) e 'Segundo minuto de rega' (valor 30). Abaixo desses campos há um campo para 'Tempo de rega (minutos)' com o valor 1 e um botão verde 'Atualizar horário de rega'. A segunda seção, intitulada 'Rega Manual', contém dois botões verdes: 'Ligar Rega Manual' e 'Desligar Rega Manual'.

(a) Interface *web* com o controle da rega, configurações e rega manual.

Esta interface é semelhante à da parte (a), mas com uma alteração no campo 'Segundo minuto de rega'. O valor inserido é 23, e uma mensagem de erro em uma caixa amarela aparece ao lado do campo, dizendo: 'O valor deve ser menor ou igual a 23.'. Uma seta vermelha aponta para o botão 'Atualizar horário de rega', indicando que a ação não foi concluída devido ao erro.

(b) Exemplo de restrição de horário.

Fonte: Desenvolvido pela autora, 2024.

As próximas funções são responsáveis por processar as configurações feitas na interface *web*.

A função (Código 22) verifica se a requisição HTTP contém os argumentos *hour*, *minute* e *time*, enviados pelo formulário HTML. Se presentes, os valores são atribuídos às variáveis de controle da irrigação e, após o processamento, o navegador é redirecionado para a página principal, exibindo as atualizações.

As outras duas funções tratam da ativação e desativação manual da irrigação (Código 23). Elas acionam as funções para ligar ou desligar o sistema e, em seguida, redirecionam o usuário para a página principal.

Logo em seguida, foi definida a função *setup*, chamada uma única vez após o microcontrolador ser ligado ou reiniciado. A diferença em relação ao Código 12 é que agora ela configura a conexão *Wi-Fi* e prepara o servidor *web*.

Como mostrado no Código 24, a função configura o *Wi-Fi* com um IP estático e tenta se conectar à rede utilizando o SSID e a senha fornecidos. Um *loop* foi adicionado para manter o sistema tentando a conexão até que seja estabelecida.

Em seguida, a função define as rotas do servidor *web* para diferentes páginas e ações, como exibir a página principal com leituras dos sensores e controles, receber

Código 22 – Função que processa as configurações de horários - terceiro código.

```
void handleSetSchedule() {
    if (server.hasArg("hour1") && server.hasArg("minute1") &&
        server.hasArg("hour2") && server.hasArg("minute2") &&
        server.hasArg("time")) {
        next_hour = server.arg("hour1").toInt();
        next_minute = server.arg("minute1").toInt();
        next_hour2 = server.arg("hour2").toInt();
        next_minute2 = server.arg("minute2").toInt();
        duracaoRega = server.arg("time").toInt() * 60000;
    }
    server.sendHeader("Location", "/");
    server.send(303);
}
```

Fonte: Elaborado pela autora, 2024.

Código 23 – Função responsável por processar as configurações manuais da rega - terceiro código.

```
void handleLigarRegaManual() {
    ligarRegaManual();
    server.sendHeader("Location", "/");
    server.send(303);
}

void handleDesligarRegaManual() {
    desligarRegaManual();
    server.sendHeader("Location", "/");
    server.send(303);
}
```

Fonte: Elaborado pela autora, 2024.

o formulário de atualização do cronograma de irrigação e as funções de ligar e desligar manualmente. Com isso configurado, o servidor *web* é iniciado.

Por fim, a função *loop* chama as funções definidas anteriormente. No Código 25, ela começa atualizando os valores dos sensores, seguida das funções para ativar a irrigação, e por fim executa a função *Handle Client*, que lida com as requisições HTTP ao servidor.

Em resumo, o código desenvolvido para a aplicação *web* implementa um sistema de irrigação que oferece ao usuário a possibilidade de monitorar os sensores conectados e controlar a irrigação, seja por meio de uma programação de horários definida previamente ou de maneira manual, conforme sua conveniência.

A interface *web* possibilita o monitoramento em tempo real dos dados fornecidos pelos sensores e permite ajustes no sistema a partir de qualquer dispositivo conectado à

Código 24 – Definição da conexão com a internet dentro da função *Setup* - terceiro código.

```
WiFi.config(ip, gateway, subnet);
WiFi.begin(ssid, password);
Serial.println("Conectando ao WiFi...");
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
}
Serial.println("WiFi conectado");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.on("/", handleRoot);
server.on("/set_schedule", handleSetSchedule);
server.on("/ligar_rega_manual", HTTP_POST, handleLigarRegaManual);
server.on("/desligar_rega_manual", HTTP_POST, handleDesligarRegaManual);
server.begin();
Serial.println("Servidor HTTP iniciado");
```

Fonte: Elaborado pela autora, 2024.

Código 25 – Função *loop* - terceiro código.

```
void loop() {
    server.handleClient();
    updateSensorsValue();
    LigarRega();
    delay(1000);
}
```

Fonte: Elaborado pela autora, 2024.

mesma rede *Wi-Fi*. Dessa forma, o usuário tem acesso a uma solução prática para controlar a irrigação. Além disso, o código integra uma função de calibração dos sensores de umidade do solo antes de o protótipo iniciar suas operações, assegurando que as medições realizadas sejam precisas.

Uma vez realizado o *upload* do código, o usuário terá acesso à interface *web* mostrada na Figura 31. Essa tela não só exibe as leituras dos sensores de maneira clara e em tempo real, mas também disponibiliza controles intuitivos que possibilitam a programação de horários para a irrigação, além da ativação e desativação manual do sistema. Essas funcionalidades proporcionam uma experiência simples e eficaz, atendendo aos objetivos de controle de irrigação com praticidade e eficiência. Ao oferecer maior flexibilidade e facilidade de uso, o sistema elaborado tem o potencial de contribuir para um manejo mais eficiente e sustentável dos recursos hídricos, podendo ser ajustado de acordo com as condições e demandas específicas de cada usuário.

Após a implementação do código e a validação do protótipo em ambiente de

Figura 31 – Interface *Web* do protótipo.

Controle do Viveiro de Mudas

Sensores
 Temperatura do ar: 25.20 °C
 Umidade do ar: 88.00 %
 Umidade do solo: 80%

Horários
 Hora atual: 0:9
 Próximas regas às: 15:26 e 15:28
 Última rega às: 0:0

Controle de Rega
 Rega ativa: Não
 Duração da rega: 60 segundos

Configuração
 Primeiro horário de rega:
 Primeiro minuto de rega:
 Segundo horário de rega:
 Segundo minuto de rega:
 Tempo de rega (segundos):

Rega Manual

Fonte: Elaborado pela autora, 2024.

teste, o sistema foi instalado no viveiro. Esse processo de instalação incluiu a montagem dos componentes, ajustes necessários e a configuração do sistema no local designado para seu funcionamento. Os detalhes dessa instalação serão apresentados na Seção 5.3.

5.3 Instalação

Para instalar o sistema no viveiro do IFMG Campus Bambuí, algumas adaptações foram necessárias. Primeiramente, foi instalada uma tomada para fornecer a conexão elétrica ao protótipo. Posteriormente, foi adicionado um fio mais longo ao sensor de umidade do solo, possibilitando seu posicionamento diretamente na terra de uma planta dentro do viveiro, garantindo a leitura precisa das condições do solo. No caso do sensor de temperatura, não houve a necessidade de extensão adicional, pois ele não precisou ser colocado muito longe para medir a temperatura ambiente de forma adequada. A Figura 32

ilustra o sensor de temperatura após a instalação.

Figura 32 – Sensor de umidade do solo após a instalação.



Fonte: Elaborado pela autora, 2024.

A caixa de controle do sistema foi fixada ao lado do painel elétrico do viveiro, aproveitando a infraestrutura elétrica já existente para facilitar a conexão e o acesso. Esse posicionamento garante a organização e a segurança dos componentes instalados.

A Figura 33 apresenta o protótipo como resultado final, após o término da instalação.

Os últimos testes foram realizados diretamente no viveiro, com monitoramento tanto pela interface *web* quanto por registros documentados, disponíveis para consulta¹.

Para simular a rotina diária de irrigação, foram programados múltiplos horários, verificando que a válvula e a bomba acionavam conforme o cronograma estabelecido. Além disso, os controles manuais foram testados com sucesso, permitindo ativar e desativar o sistema pelos botões específicos, demonstrando plena funcionalidade.

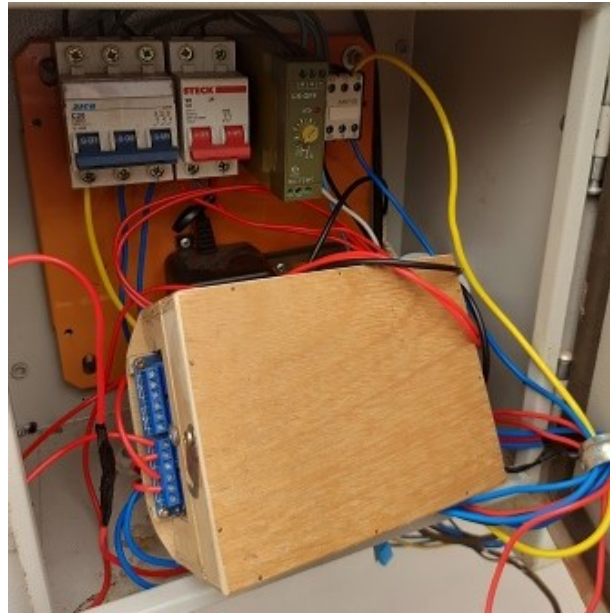
Durante a execução no viveiro, observou-se uma limitação na conexão de internet, que apresentava instabilidade. Para contornar o problema, foi necessário utilizar um celular para rotear a internet e garantir a comunicação do sistema.

A Figura 34 ilustra o momento dos testes, com um *notebook* acessando a interface *web*, monitorando os dados dos sensores e acionando os controles de irrigação.

Após a instalação e os testes realizados, o sistema demonstrou ser uma solução viável e eficaz para a automação da irrigação no viveiro. O protótipo funcionou

¹ https://drive.google.com/drive/folders/1-QbBZDOIzyW9k-S2QWEGPU2xNb2-sx_n

Figura 33 – Protótipo instalado no painel elétrico.



Fonte: Elaborado pela autora, 2024.

Figura 34 – Utilização de um *notebook* para acessar a interface *web* e realizar os testes necessários.



Fonte: Elaborado pela autora, 2024.

conforme esperado, ativando e desativando a válvula e a bomba de acordo com os horários programados, além de responder com precisão aos comandos manuais na interface *web*.

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um sistema automatizado de irrigação para o viveiro do IFMG Campus Bambuí, projetado para oferecer uma solução prática e acessível que não apenas monitora variáveis ambientais como temperatura e umidade do solo e do ar, mas também controla a irrigação de maneira automatizada conforme horários programados. O sistema visa maximizar a eficiência no uso de recursos, minimizando desperdício de água e energia elétrica, além de reduzir a necessidade de mão de obra para a rega manual.

Com base nas observações e necessidades identificadas nas visitas iniciais ao viveiro, foi implementado um protótipo adaptável para atender a essas demandas. Durante o processo de instalação e testes, o sistema demonstrou sua eficácia ao cumprir a rotina de irrigação planejada e executar os comandos configurados. A capacidade de monitoramento e controle por meio de um dispositivo móvel permite que ajustes sejam feitos de forma visualmente prática.

Na escolha dos componentes, o ESP32 foi selecionado como microcontrolador principal, destacando-se pela integração eficiente com sensores e atuadores, além de sua conectividade Wi-Fi ele proporcionou flexibilidade para adaptações futuras e uma interface de controle acessível via *web*. Essa arquitetura foi projetada de maneira modular, facilitando a manutenção e a expansão do sistema conforme novas necessidades surgirem, como a inclusão de novos setores de irrigação ou sensores adicionais.

Além de contribuir para o campo de sistemas embarcados aplicados à agricultura, o projeto serve como uma base escalável e replicável para outras áreas agrícolas que buscam aumentar a eficiência e sustentabilidade de seus processos. Dessa forma, ele propicia uma aplicação direta dos conceitos de agricultura de precisão e gestão inteligente dos recursos hídricos, alinhando-se às práticas modernas de sustentabilidade e inovação na agricultura.

6.1 Trabalhos Futuros

Algumas sugestões de trabalhos futuros incluem:

- Controle da irrigação com base nos dados fornecidos pelos sensores, ajustando automaticamente os processos de rega conforme os parâmetros ambientais monitorados.
- Implementar um *software* para o monitoramento e controle remoto do sistema de irrigação, permitindo que os responsáveis pelo viveiro possam ajustar as configurações de irrigação à distância.
- Expandir o sistema de irrigação para os demais setores do viveiro, proporcionando uma irrigação mais eficiente e precisa em todo o viveiro.
- Aumentar a flexibilidade de dias e quantidade de dias pra rega.

REFERÊNCIAS

- ADEBAYO, A. A. *et al.* Smart Agriculture for Sustainability: The Implementation of Smart Irrigation Using Real-Time Embedded System Technology. **IEEE Conference Publication**, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10548972>.
- ALEXANDER, C. K. .; SADIKU, M. N. O. **Fundamentals os Electric Circuits**. 7^a: McGraw-Hill Education, 2021.
- ALJUNDI, L. **Using the Arduino Software (IDE)**: The offline IDE makes it easy to write code and upload it to the board without an Internet connection. 2023. Disponível em: <https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide>. Acesso em: 26 mai. 2023.
- BARRETT, S. F. **Arduino microcontroller processing for everyone!** Springer Nature, 2022.
- BASSO, B.; ANTLE, J. Digital agriculture to design sustainable agricultural systems. **Nature Sustainability**, Nature Publishing Group, v. 3, n. 4, p. 254–256, 2020.
- BRAGA, N. C. **Relés: Circuitos e aplicações**. Editora Newton C. Braga, 2017.
- CIRCUITSTODAY. **Draw Circuits and Simulate Online Using EasyEDA**. 2020. Acesso em: 21 de maio de 2024. Disponível em: <https://www.circuitstoday.com/draw-circuits-and-simulate-online-using-easyeda>.
- DEOGIRIKAR, A. A. **Protected Cultivation and Secondary Agriculture: B.Sc. Agri. (Hons.) Third Year - VI Semester AGRICULTURAL ENGINEERING Question and Answer as per Exam Format**. New Delhi: The V Dean's Committee of ICAR, 2021.
- DIDI, Z.; EL AZAMI, I. IoT, Comparative Study Between the Use of Arduino Uno, Esp32, and Raspberry pi in Greenhouses. In: SPRINGER. **INTERNATIONAL Conference on Digital Technologies and Applications** [...]. 2022. P. 718–726.
- DINIZ, A. M. **Sistema automatizado de aquisição, em tempo real, de umidade e temperatura do solo na irrigação**. 2017. Dissertação (Pós Graduação em Engenharia Agrícola) – Universidade estadual do oeste do Paraná, Cascavel, 2017.
- ESPRESSIF SYSTEMS. **ESP32 Series Technical Reference Manual**. 2022. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.
- FACHINELLO, J. C.; HOFFMANN, A.; NACHTIGAL, J. C. *et al.* **Propagação de plantas frutíferas**. Embrapa informação tecnológica Brasília, 2005.
- FERNANDES, D. G. **Sistema automatizado de controle de estufas para cultivo de hortaliças**. 2017.

FERREIRA, G. S.; FAGUNDES, F. D. **CONSTRUÇÃO DE UMA UNIDADE RE-PROGRAMÁVEL DE EFEITO TREMOLO PARA GUITARRA ELÉTRICA.**

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa.** Plageder, 2009.

GOMES, R. S.; LIMA, P. T. Desafios e Eficácia da Irrigação Manual em Ambientes Controlados de Estufas Agrícolas. **Revista Brasileira de Agricultura Sustentável**, Sociedade Brasileira de Agricultura Sustentável, v. 15, n. 2, p. 101–115, 2023. DOI: 10.1007/s4056-015-1234-5.

HARVIE, L. **Low Power Design Tips & Techniques for Embedded Systems.** 2023. Available at <https://www.runtimerec.com/low-power-design-tips-techniques-for-embedded-systems>.

HEMACHANDRA, R. B. What is an Embedded System? Complete Guide. In: MAVEN TECH PUBLISHING. EMBEDDED Systems Conference [...]. 2023. P. 320–326.

HENRIQUES, L. F. A. *et al.* **Implementação e monitoramento de um sistema de irrigação automatizado em IoT utilizando módulo ESP32 em plantio caseiro.** Brasil, 2021.

JUNIOR, V. d. S. Microcontroladores PIC 16F e 18F: Teoria e Prática. **São Paulo: Instituto Newton C Braga**, 2013.

KERSCHBAUMER, R. Engenharia de controle e Automação-Microcontroladores. [link](#)]. **Acesso em**, v. 9, 2021.

KURNIAWAN, A. **Internet of Things with ESP32.** 1ª: Packt Publishing, 2019.

LACERDA, D. P. *et al.* Design Science Research: método de pesquisa para a engenharia de produção. **Gestão & produção**, SciELO Brasil, v. 20, p. 741–761, 2013.

LIMA PEREIRA, F. A. de *et al.* DESENVOLVIMENTO DE SENSOR DE UMIDADE DO SOLO UTILIZANDO O PRINCÍPIO DA RESISTÊNCIA ELÉTRICA. **IRRIGA**, v. 26, n. 1, p. 29–41, 2021.

MALLAREDDY, M. *et al.* Maximizing water use efficiency in rice farming: A comprehensive review of innovative irrigation management technologies. **Water**, MDPI, v. 15, n. 10, p. 1802, 2023.

MARQUES FILHO, A. C. **Sistema de automação e controle inteligente para cultivo protegido.** 2017.

MCROBERTS, M. **Arduino básico.** Novatec Editora, 2018.

MENDES, L. C.; PEREIRA, R. T. **Sistemas Inteligentes de Irrigação: Inovações e Sustentabilidade na Agricultura.** 1. ed. São Paulo, Brasil: Editora AgroScience, 2023.

NEWTON, C. B. **Eletrônica Básica.** (2001–1ª ed.) São Paulo: Editora do Instituto Newton C. Braga, 2012.

OLIVEIRA, F. R.; SOUZA, B. L. A Influência da Irrigação Automatizada na Eficiência Hídrica e Produtividade em Estufas. **Ciência e Tecnologia Agrícola**, Editora Científica Rural, v. 9, n. 4, p. 312–327, 2021. DOI: 10.1007/s4018-021-1154-3.

PEREIRA, F. **Reles Eletromecânicos**. Disponível em: https://www.voltimum.com.br/sites/www.voltimum.com.br/files/pdflibrary/whitepaper_br_reles_eletromecanicos_0.pdf.

RAVI, S. **How to Use DS1307 RTC with Arduino**. 2022.

REESE, R. B.; BRUCE, J. W.; A., J. B. **Microcontrollers: From Assembly Language to C Using the PIC24 Family**. 2ª: Cengage Learning, 2014.

REIS, N. V. dos *et al.* **Construção de estufas para produção de hortaliças nas Regiões Norte, Nordeste e Centro-Oeste**. 2005.

RODYCZ, H. M.; VIEIRA, M. E. Sistema Móvel de Captação de Água Controlado Remotamente. **Seminário de Eletrônica e Automação de Ponta Grossa**, 2015.

RUSH, P. Proteção e automação de redes: conceito e aplicação. **São Paulo**, 2011.

SALACHNA, P.; WILAS, J.; ZAWADZIŃSKA, A. The effect of chitosan coating of bulbs on the growth and flowering of *Ornithogalum saundersiae*. In: XXIX International Horticultural Congress on Horticulture: Sustaining Lives, Livelihoods and Landscapes (IHC2014) [...]. Szczecin, Poland, 2014. P. 115–118.

SAMIULLAH, M.; IRFAN, M. Z.; RAFIQUE, A. Microcontrollers: A Comprehensive Overview and Comparative Analysis of Diverse Types. **National University of Sciences and Technology (NUST)**, Engineering Archive, 2023.

SANTOS, C. A. e. a. Consumo hídrico de manjerição, hortelã e orégano em diferentes estádios fenológicos. **Revista Brasileira de Engenharia Agrícola e Ambiental**, v. 25, n. 4, p. 324–329, 2021.

SANTOS, C. A.; OLIVEIRA, F. R. Sistemas de Irrigação Automatizada: Tecnologias e Desafios na Agricultura Moderna. **Revista Brasileira de Agricultura**, Associação Brasileira de Agricultura, v. 12, n. 4, p. 45–58, 2022. DOI: 10.1234/rba.v12n4.4567.

SAVAGE, L. **Master of the Flow: A Complete Guide to Modern Irrigation Techniques**. Independently Published, 2023. ISBN 9798857162248.

SAXENA, P. **Arduino Sensors: A Beginner's Guide to Using Sensors with Arduino**. 1ª: Packt Publishing, 2018.

SCHUBERT, K. A. **Proposta de otimização de comunicação para plataformas microcontroladas utilizando mapeamento em memória externa**. 2020.

SILVA, A. P.; CARVALHO, J. R. **Sistemas de Irrigação Inteligente e Automação em Estufas Agrícolas**. 1. ed. São Paulo, Brasil: Editora Agrotech, 2023.

- SILVA, A. O.; LOURENÇO, L. H.; NASCIMENTO, M. d. O. **Sistema para avaliação da umidade do solo visando automatização de irrigação**. 2022.
- SOUSA, C. R. de; SILVA NASCIMENTO, G. A. da; SANTOS, M. P. dos. EXPERIÊNCIA NA CONSTRUÇÃO DE UM SISTEMA DE IRRIGAÇÃO AUTOMATIZADO DE UMA HORTA NO ENSINO MÉDIO TÉCNICO. **Scientia: Revista Científica Multidisciplinar**, v. 7, n. 2, p. 77–95, 2022.
- SOUZA, G. V. P. d. **Desenvolvimento de um sistema de comunicação IoT para gerenciamento energético de casas inteligentes**. 2022. B.S. thesis – Universidade Tecnológica Federal do Paraná.
- STROSKI, P. N. **Conheça o relé eletromecânico**. 2018. Disponível em: <https://www.electricalibrary.com/2018/08/06/conheca-o-rele-eletromecanico/>. Acesso em: 2024.
- THOMAZINI, D.; ALBUQUERQUE, P. U. B. de. **Sensores industriais: fundamentos e aplicações**. Saraiva Educação SA, 2020.
- TREMPER, D. P. **Irrigação em paisagismo**. 2015.
- WANG, X. *et al.* Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies. **Journal of the IEEE**, v. 8, p. 395–420, 2021.
- WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. Rio de Janeiro: Elsevier, 2009. v. 3, p. 17–24.
- WENDLING, I.; FERRARI, M. P.; GROSSI, F. **Curso intensivo de viveiros e produção de mudas**. 2002.
- WENDLING, M. Sensores. **Universidade Estadual Paulista. São Paulo**, v. 2010, p. 20, 2010.

APÊNDICE A - Código do projeto de Interface *Web* para Controle da Irrigação

```
//-----Bibliotecas-----

#include <RTCLib.h>
#include <DHT.h>
#include <WiFi.h>
#include <WebServer.h>

//-----Definição dos pinos-----

#define RELE1_PIN 27
#define RELE2_PIN 26
#define RELE3_PIN 25
#define RELE4_PIN 32

// Pino analógico para o sensor de umidade do solo
#define SOIL_HUMIDITY_PIN A0
// Pino ao qual o sensor DHT11 está conectado
#define DHT_PIN 14

// Tipo de sensor DHT
#define DHT_TYPE DHT11
DHT dht(DHT_PIN, DHT_TYPE);
RTC_DS1307 rtc;

//-----Conexão na rede local sem fio-----

const char* ssid = "IFMG-Bambui-Convitados";
const char* password = "2024Bambui1FMG";

IPAddress ip(10, 60, 17, 50);
IPAddress gateway(10, 60, 16, 1); // Endereço do gateway
IPAddress subnet(255, 255, 252, 0); // Máscara de sub-rede

WebServer server(80);

//-----Variáveis globais-----

int current_hour; // Será a hora de acordo com o rtc
int current_minute; // Variável para armazenar o minuto atual
int aux = 0; // usado
a na calibração
```

```

//Valores atuais recebido dos sensores
float soil_humidity = 0;
float air_temperature = 0;
float air_humidity = 0;

int leituraSeca = 0;
int leituraUmida = 0;

//horários
int last_hour = 0; //será atualizado quando a rega acontecer
int last_minute = 0;
int next_hour = 10; //será definido no site
int next_minute = 00;
int next_hour2 = 15;
int next_minute2 = 30;

// Variável para controlar se a rega está ativa
bool regaAtiva = false;
// Armazena o tempo em que a rega foi iniciada
unsigned long tempoInicioRega;
// Duração da rega em milissegundos
unsigned long duracaoRega = 300000;

//-----função que controla a rega de acordo com as horas-----

void LigarRega() {
  Serial.print("Verificando condição de rega: ");
  Serial.print(current_hour);
  Serial.print(":");
  Serial.print(current_minute);
  Serial.print(" - ");
  Serial.print(next_hour);
  Serial.print(":");
  Serial.println(next_minute);

  // Verifica se esta na hora de ligar a rega
  if ((current_hour == next_hour && current_minute == next_minute ||
       current_hour == next_hour2 && current_minute == next_minute2) &&
      !regaAtiva) {
    Serial.println("Rega ligada");
    digitalWrite(RELE1_PIN, LOW);
    delay(2000);
    digitalWrite(RELE2_PIN, LOW);
    tempoInicioRega = millis(); // Registra o tempo de início da rega
    regaAtiva = true;
    last_hour = current_hour;
  }
}

```

```

    last_minute = current_minute;
}

// Verifica se a rega deve ser desligada após a duração especificada
if (regaAtiva && millis() - tempoInicioRega >= duracaoRega) {
    Serial.println("Rega desligada");
    digitalWrite(RELE2_PIN, HIGH);
    delay(2000);
    digitalWrite(RELE1_PIN, HIGH);
    regaAtiva = false;
}
}

//-----Funções de controle manual-----

void ligarRegaManual() {
    Serial.println("Rega manual ligada");
    digitalWrite(RELE1_PIN, LOW);
    delay(2000);
    digitalWrite(RELE2_PIN, LOW);
    regaAtiva = true;
    tempoInicioRega = millis();
    last_hour = current_hour;
    last_minute = current_minute;
}

void desligarRegaManual() {
    Serial.println("Rega manual desligada");
    digitalWrite(RELE2_PIN, HIGH);
    delay(2000);
    digitalWrite(RELE1_PIN, HIGH);
    regaAtiva = false;
}

//-----Calibrando os sensores-----

void calibrarSensorUmidadeSolo() {
    if (aux == 0){
        Serial.println("Insira o sensor em solo seco e aguarde...");
        delay(10000);
        leituraSeca = analogRead(SOIL_HUMIDITY_PIN);
        Serial.println("Leitura seca concluida");
        delay(1000);
        Serial.println("Insira o sensor em solo úmido e aguarde...");
        delay(10000);
    }
}

```

```

        leituraUmida = analogRead(SOIL_HUMIDITY_PIN);
        Serial.println("Leitura umida concluida");
        delay(1000);
        Serial.println("Calibração concluída.");
        aux = 1;
    }
}

//-----Atualiza os valores dos sensores-----

void updateSensorsValue() {
    air_humidity = dht.readHumidity();
    air_temperature = dht.readTemperature();
    soil_humidity = analogRead(SOIL_HUMIDITY_PIN);

    if (isnan(air_humidity) || isnan(air_temperature)) {
        Serial.println("Falha na leitura do DHT!");
        return;
    }

    DateTime now = rtc.now();
    current_hour = now.hour();
    current_minute = now.minute();

    //Conversão do valor do sensor yl-69 para umidade percentual
    soil_humidity = map(soil_humidity, leituraSeca, leituraUmida, 0, 100);
    Serial.print("Temp: ");
    Serial.print(air_temperature);
    Serial.print(" Hum: ");
    Serial.print(air_humidity);
    Serial.print(" Soil: ");
    Serial.println(soil_humidity);
}

//-----função que conecta na internet-----

void handleRoot() {
    updateSensorsValue(); // Atualiza os valores dos sensores

    // Inicia a criação da página HTML
    String page = "<html><head><title>Controle do Viveiro</title>";
    // Define a codificação de caracteres para UTF-8
    page += "<meta charset='UTF-8'>";
}

```



```

page += "<style>"; // Inicia a seção de estilo CSS
page += "body { font-family: Arial, sans-serif; margin: 0;
        padding: 0; }"; // Estilo para o corpo da página
// Estilo para o contêiner da página
page += ".container { padding: 20px; }";

page += "h1 { background-color: #4CAF50; color: white;
        padding: 10px; }"; // Estilo para o cabeçalho
page += ".section { border: 1px solid black; padding: 10px;
        margin-bottom: 20px; }"; // Estilo para as seções
// Estilo para os subtítulos das seções
page += ".section h2 { margin-top: 0; }";
page += "input[type='submit'] { background-color: #4CAF50; color:
        white; border: none; padding: 10px 20px; cursor: pointer;
        border-radius: 12px; }"; // Estilo para os botões de envio
// Estilo para os botões de envio quando o mouse está sobre eles
page += "input[type='submit']:hover { background-color: #45a049; }";
page += "</style>";
page += "</head><body><div class='container'>";
page += "<h1>Controle do Viveiro de Mudas</h1>";

// Seção de sensores
page += "<div class='section'>";
page += "<h2>Sensores</h2>";
page += "<p>Temperatura do ar: " + String(air_temperature) + " °C</p>";
page += "<p>Umidade do ar: " + String(air_humidity) + " %</p>";
page += "<p>Umidade do solo: " + String(soil_humidity) + " %</p>";
page += "</div>";

// Seção de horários
page += "<div class='section'>";
page += "<h2>Horários</h2>";
page += "<p>Hora atual: " + String(current_hour) + ":" +
        String(current_minute) + "</p>";
page += "<p>Horário das regas: " + String(next_hour) + ":" +
        String(next_minute) + " e " + String(next_hour2) + ":" +
        String(next_minute2) + "</p>";
page += "<p>Horário da última rega: " + String(last_hour) + ":" +
        String(last_minute) + "</p>";
page += "</div>";

// Seção de controle de rega
page += "<div class='section'>";
page += "<h2>Controle de Rega</h2>";
page += "<p>Rega ativa: " + String(regaAtiva ? "Sim" : "Não") +
        "</p>";

```

```

page += "<p>Duração da rega: " + String(duracaoRega / 60000) +
        " minutos</p>";
page += "</div>";

// Seção de configuração
page += "<div class='section'>";
page += "<h2>Configuração</h2>";
// Formulário para definir o próximo horário de rega
page += "<form action='/set_schedule' method='GET'>";
page += "Primeiro horário de rega: <input type='number' name='hour1'
        min='0' max='23' value='" + String(next_hour) + "'><br>";
page += "Primeiro minuto de rega: <input type='number' name='minute1'
        min='0' max='59' value='" + String(next_minute) + "'><br>";
page += "Segundo horário de rega: <input type='number' name='hour2'
        min='0' max='23' value='" + String(next_hour2) + "'><br>";
page += "Segundo minuto de rega: <input type='number' name='minute2'
        min='0' max='59' value='" + String(next_minute2) + "'><br>";
page += "Tempo de rega (minutos): <input type='number' name='time'
        min='1' value='" + String(duracaoRega / 60000) + "'><br>";
page += "<input type='submit' value='Atualizar horário de rega'>";
page += "</form>";
page += "</div>";

// Seção de controle manual
page += "<div class='section'>";
page += "<h2>Rega Manual</h2>";
page += "<form action='/ligar_rega_manual' method='POST'>";
page += "<input type='submit' value='Ligar Rega Manual'>";
page += "</form>";
page += "<form action='/desligar_rega_manual' method='POST'>";
page += "<input type='submit' value='Desligar Rega Manual'>";
page += "</form>";
page += "</div>";
page += "</div></body></html>";
// Envia a página HTML para o cliente
server.send(200, "text/html", page);
}

//-----função que atualiza o horário de rega-----

void handleSetSchedule() {
    if (server.hasArg("hour1") && server.hasArg("minute1") &&
        server.hasArg("hour2") && server.hasArg("minute2") &&
        server.hasArg("time")) {
        next_hour = server.arg("hour1").toInt();
        next_minute = server.arg("minute1").toInt();
    }
}

```

```

        next_hour2 = server.arg("hour2").toInt();
        next_minute2 = server.arg("minute2").toInt();
        duracaoRega = server.arg("time").toInt() * 60000;
    }
    server.setHeader("Location", "/");
    server.send(303);
}

//-----função que controla a função manual da rega-----

void handleLigarRegaManual() {
    ligarRegaManual();
    server.setHeader("Location", "/");
    server.send(303);
}

void handleDesligarRegaManual() {
    desligarRegaManual();
    server.setHeader("Location", "/");
    server.send(303);
}

//-----setup-----

void setup() {
    Serial.begin(115200);
    //-----Definição os pinos-----

    pinMode(RELE1_PIN, OUTPUT);
    pinMode(RELE2_PIN, OUTPUT);
    pinMode(RELE3_PIN, OUTPUT);
    pinMode(RELE4_PIN, OUTPUT);
    pinMode(SOIL_HUMIDITY_PIN, INPUT);
    digitalWrite(RELE1_PIN, HIGH);
    digitalWrite(RELE2_PIN, HIGH);
    digitalWrite(RELE3_PIN, HIGH);
    digitalWrite(RELE4_PIN, HIGH);

    calibrarSensorUmidadeSolo();

    //-----Inicialização DHT e RTC-----

    Serial.println("Inicializando o sensor DHT...");
    dht.begin(); // Inicializa o sensor DHT
    delay(2000);

```

```

if (!rtc.begin()) {
    Serial.println("Erro ao iniciar o RTC!");
    while (1);
}

//-----Conexão WIFI-----

WiFi.config(ip, gateway, subnet);
WiFi.begin(ssid, password);

Serial.println("Conectando ao WiFi...");
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
}

Serial.println();
Serial.println("WiFi conectado");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

server.on("/", handleRoot);
server.on("/set_schedule", handleSetSchedule);
server.on("/ligar_rega_manual", HTTP_POST, handleLigarRegaManual);
server.on("/desligar_rega_manual", HTTP_POST, handleDesligarRegaManual);
server.begin();
Serial.println("Servidor HTTP iniciado");
}

//-----loop-----

void loop() {
    server.handleClient();
    updateSensorsValue();
    LigarRega();

    delay(1000);
}

```