UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS AGRÁRIAS
DEPARTAMENTO DE ENGENHARIA AGRÍCOLA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA AGRÍCOLA

TALYSON WEBER RODRIGUES ROLIM

UNIDADE AUTÔNOMA DE MANEJO DA IRRIGAÇÃO DE BAIXO CUSTO CONTROLADO VIA APLICATIVO PARA CULTIVOS EM AMBIENTE PROTEGIDO

FORTALEZA
2021

TALYSON WEBER RODRIGUES ROLIM

UNIDADE AUTÔNOMA DE MANEJO DA IRRIGAÇÃO DE BAIXO CUSTO CONTROLADO VIA APLICATIVO PARA CULTIVOS EM AMBIENTE PROTEGIDO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Agrícola do Departamento de Engenharia Agrícola da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Engenharia Agrícola. Área de concentração: Engenharia de Irrigação.

Orientador: Prof. Dr. Alexsandro Oliveira da Silva.

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R654u Rolim, Talyson Weber Rodrigues.

Unidade autônoma de manejo da irrigação de baixo custo controlado via aplicativo para cultivo em ambiente protegido / Talyson Weber Rodrigues Rolim. – 2021.

140 f.: il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências Agrárias, Programa de Pós-Graduação em Engenharia Agrícola, Fortaleza, 2021.

Orientação: Prof. Dr. Alexsandro Oliveira da Silva.

1. Automação. 2. Ambiente protegido. 3. Internet das Coisas. I. Título.

CDD 630

TALYSON WEBER RODRIGUES ROLIM

UNIDADE AUTÔNOMA DE MANEJO DA IRRIGAÇÃO DE BAIXO CUSTO CONTROLADO VIA APLICATIVO PARA CULTIVOS EM AMBIENTE PROTEGIDO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Agrícola do Departamento de Engenharia Agrícola da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Engenharia Agrícola. Área de concentração: Engenharia de Irrigação.

Aprovada em ____/____/_____

BANCA EXAMINADORA

_____

Prof. Dr. Alexsandro Oliveira da Silva (Orientador).
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Renato Silvio da Frota Ribeiro
Universidade Federal do Ceara (UFC)

_____

Prof. Dr. Miguel Júlio Machado Guimarães
Instituto Federal do Maranhão (IFMA)

_____

Dr. Henrique Oldoni
Universidade Estadual de Campinas (UNICAMP)

A Deus.

Aos meus pais, irmãos e amigos.

AGRADECIMENTOS

A Deus, por permitir concluir mais essa etapa e proporcionar esse momento.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo apoio financeiro com a manutenção da bolsa de auxílio. O presente trabalho foi realizado com apoio do CNPq, por meio do PROAP, que também agradeço. Como também a UFC pela oportunidade de participar do programa de pós-graduação e crescer pessoal e profissionalmente.

Ao Prof. Dr. Alexsandro Oliveira da Silva, pelo apoio, orientação e oportunidade de aprendizado que me ofereceu. Aos professores participantes da banca examinadora prof. Renato da Frota Ribeiro, prof. Henrique Oldoni, prof. Miguel Júlio Machado Guimarães, pelo tempo, pelas valiosas colaborações e sugestões.

Aos amigos que o mestrado me ofereceu: Bruna Aires, Jenyffer Gomes e Albano Uchoa, obrigado pelas conversas, pelo apoio e por toda parceria dentro e fora do âmbito universitário. Também agradeço a Letícia, Glauco e Weverton por todo apoio e ajuda durante o experimento e também a Yngrid que me ajudou todos os dias com seu apoio e incentivo, que foram fundamentais para meu sucesso.

Aos meus pais Lyryal e Hortência que são meu alicerce e também flores do meu jardim, que como as rosas, proporcionaram em minha vida os momentos de ternura simbolizados pelo aroma, como também as lições que só os espinhos podem dar. Todos os momentos de risos e lágrimas que me ensinaram os lados da vida, sejam nos momentos bons ou ruins e olhar pra cima sempre com o pensamento de gratidão.

Ao meu padrinho e segundo pai, Paulo Régis, que me acolheu em sua residência e me tomou como filho quando vim morar em fortaleza, sempre cuidando de mim como pôde e que sempre estarei em dívida e espero um dia conseguir retribuir tudo que me proporcionou.

Aos meus familiares e amigos que me incentivaram a seguir em frente e ajudaram na formação de caráter e profissional, assim como todos meus professores desde a infância até a universidade. Meu muito obrigado.

“O que temos, nós deixamos. O que somos, nós levamos.” Divaldo Franco.

RESUMO

Diante dos constantes aumentos de custos de água, além da quantidade limitada desse recurso, a automação de sistemas de irrigação pode auxiliar os produtores a reduzir os gastos e elevar a produção, consequentemente, a lucratividade. Diante disso, o objetivo desta pesquisa foi desenvolver e testar um protótipo de sistema autônomo de irrigação com baixo custo para o manejo da irrigação na produção de hortaliças, gerando economia de mão de obra, analisando o seu desempenho e sua implicação econômica. O experimento foi realizado em casa de vegetação do Departamento de Engenharia Agrícola – DENA, localizada na estação meteorológica da Universidade Federal do Ceará – UFC, campus do PICI, em Fortaleza – CE. A cultura utilizada foi a rúcula (*Eruca Sativa* L), cultivada em dois ciclos de produção no período compreendido entre 17 de dezembro de 2020 à 16 de janeiro de 2021 e entre 1 de fevereiro a 3 de março de 2021. O experimento foi conduzido em blocos casualizados com 4 tipos de manejo da irrigação, dividido em automáticos e manuais: manejo automático via solo (IAS); manejo automático via clima (IAC), manejo manual via solo (IMS) e manejo manual via clima (IMC). Cada tratamento possuía 8 repetições, totalizando 32 parcelas experimentais. Foram obtidos dados de crescimento como altura das plantas, número de folhas e dados de produção como massa fresca e seca da parte aérea. Foi utilizada análise estatística, pelo teste F (Anova) para os fatores estudados e quando estes foram significativos, realizou-se o teste de Tukey ambos a 5% de probabilidade. Para a variável massa fresca, observou-se que os tratamentos IAC (17,75 g planta^{-1}), IAS (12,38 g planta^{-1}) e IMC (8,63 g planta^{-1}) no ciclo 1 foram estatisticamente semelhantes entre si pelo teste de Tukey a 5% de probabilidade, já no ciclo 2 apenas os tratamentos IAC (16,29 g planta^{-1}) e IAS (19,80 g planta^{-1}) tiveram essa semelhança estatística. Os tratamentos IAC e IAS obtiveram os melhores resultados tanto no ciclo 1 como no segundo. Ambos os manejos (IAS e IAC) são recomendados com base nesta pesquisa, contudo, considerando dificuldades financeiras do pequeno produtor o IMC pode ser considerado uma opção desejável em condições econômicas não favoráveis.

Palavras-chave: automação; internet das coisas; ambiente protegido.

ABSTRACT

Given the constant increases in water costs, in addition to the limited amount of this resource, the automation of irrigation systems can help producers reduce expenses and increase production, consequently, profitability. Therefore, the objective of this research was to develop and test a prototype of an autonomous irrigation system with low cost for the management of irrigation in the production of vegetables, generating labor savings, analyzing its performance and its economic implications. The experiment was carried out in a greenhouse of the Department of Agricultural Engineering - DENA, located at the meteorological station of the Federal University of Ceará - UFC, PICI campus, in Fortaleza - CE. The culture used was rocket (Eruca Sativa L), cultivated in two production cycles in the period between December 17, 2020 and January 16, 2021 and between February 1 and March 3, 2021. The experiment was carried out in randomized blocks with 4 types of irrigation management, divided into automatic and manual: automatic soil management (IAS); automatic handling via climate (IAC), manual handling via soil (IMS) and manual handling via climate (IMC). Each treatment had 8 repetitions, totaling 32 experimental plots. Growth data such as plant height, number of leaves and production data such as fresh and dry mass of shoots were collected. Statistical analysis was used by the F test (ANOVA) for the studied factors and when they were evaluated, the Tukey test was performed, both at 5% probability. For the variable fresh mass, it was observed that the treatments IAC (17.75 g plant-1), IAS (12.38 g plant-1) and BMI (8.63 g plant-1) in cycle 1 were statistically related between each other by the Tukey test at 5% probability, in cycle 2 only the treatments IAC (16.29 g plant-1) and IAS (19.80 g plant-1) had this statistical similarity. The IAC and IAS treatments obtained the best results both in cycle 1 and in the second. Both managements (IAS and IAC) are recommended based on this research, however, considering the financial difficulties of the small producer, BMI can be considered a desirable option in unfavorable economic conditions.

Keywords: automation; internet of things; protected environment.

LISTA DE ILUSTRAÇÕES

Figura 1 – Localização da casa de vegetação 22

Figura 2 – Dados climáticos no interior do ambiente protegido: temperatura e umidade do ar para o ciclo 1 (A) e para o ciclo 2 (B)..... 23

Figura 3 – Visão geral do delineamento experimental e disposição dos tratamentos 25

Figura 4 – Curva característica do solo obtida para realização do manejo da irrigação 26

Figura 5 – Tanque classe A utilizado no experimento 27

Figura 6 – Organograma de funcionamento do dispositivo IAS (A) e interface de comando no aplicativo do dispositivo IAS (B) 28

Figura 7 – Sensor de umidade utilizado e armazenamento de dados (A), válvula solenóide instalada (B), aplicativo móvel (C) e fita gotejadora (D) 30

Figura 8 – Organograma de funcionamento do dispositivo IAC (A) e design de interface do aplicativo 31

Figura 9 – Componentes do manejo autônomo de irrigação via clima utilizado no experimento 32

Figura 10 – Lâmina aplicada no ciclo 1 e ciclo 2 em milímetro 36

Figura 11 – Temperaturas máxima, média e mínima (A) e radiação solar (B) durante o ciclo 1 de produção e (C) e (D), para o ciclo 2 respectivamente 38

Figura 12 – Potencial matricial e lâmina de irrigação acumulada no 1° ciclo (A) e 2° ciclo (B) 40

Figura 13 – Nível do tanque classe A e lâmina aplicada no ciclo 1 (A) e no ciclo 2 (B) 42

Figura 14 – Altura de plantas submetida ao teste de Tukey aos 10 DAT, DMS=2,78 (A); 20 DAT, DMS = 5,88 (B); 30 DAT, DMS = 8,37 (C) no ciclo 1 44

Figura 15 – Altura de plantas submetida ao teste de Tukey aos 20 DAT, DMS = 3,21 (A) e 30 DAT, DMS = 3,63 (B) no ciclo 2 da cultura da rúcula sob diferentes manejos de irrigação 44

Figura 16 – Massa fresca de rúcula submetida a diferentes manejos de irrigação no ciclo 1 de produção para o ciclo 1 (A) e para o ciclo 2 (B) 49

Figura 17 – Variáveis de rendimento para área foliar no primeiro (DMS = 147,78) (A) e ciclo 2 (DMS = 88,98) (C), massa fresca da parte aérea no primeiro (DMS =

12,70) (B) e segundo (DMS= 7,01) (D) ciclo para a cultura da rúcula sob diferentes manejos de irrigação 49

Figura 18 – Eficiência do uso da água para a cultura da rúcula (DMS = 2,37) no ciclo 1 de produção sob diferentes manejos de irrigação 50

LISTA DE TABELAS

Tabela 1 – Características química de solo utilizado no experimento 23

Tabela 2 – Dados de irrigação do sistema automático via solo 37

Tabela 3 – Dados de irrigação do sistema automático via clima durante o experimento 39

Tabela 4 – Dados médios de irrigação do sistema manual via solo durante o experimento 41

Tabela 5 – Dados de irrigação do sistema manual via clima durante o experimento .. 41

Tabela 6 – Análise e variância (quadrado médio) para a variável altura de plantas aos 10, 20 e 30 dias após o transplantio (DAT) 43

Tabela 7 – Análise e variância (quadrado médio) para a variável número de folha aos 10, 20 e 30 dias após o transplantio (DAT) 46

Tabela 8 – Análise e variância (quadrado médio) para a variável produtividade por **quantidade de água utilizada** 50

SUMÁRIO

1 INTRODUÇÃO..... 14
2 HIPÓTESE E OBJETIVOS..... 16
2.1 Hipótese..... 16
2.2 Objetivo geral..... 16
2.3 Objetivos específicos..... 16
3 REVISÃO DE LITERATURA..... 17
3.1 Tecnologia no desenvolvimento agrícola: agricultura digital..... 17
3.2 Automação na agricultura irrigada..... 18
3.3 Utilização de sensores na agricultura..... 19
3.4 Cultivo em ambiente protegido..... 20
3.5 Método de controle IoT..... 20
4 MATERIAIS E MÉTODOS..... 22
4.1 Caracterização experimental..... 22
4.2 Cultura utilizada no experimento..... 24
4.3 Tratamentos e delineamento experimental..... 24
4.3.1 Manejo de irrigação manual via solo..... 25
4.3.2 Manejo de irrigação manual via clima..... 26
4.3.3 Manejo da irrigação automática via solo..... 27
4.3.4 Manejo da irrigação automática via clima..... 30
4.4 Variáveis Analisadas..... 34
4.4.1 Produtividade das culturas..... 34
4.4.2 Custos de mão de obra..... 34
4.4.3 Utilização de recursos..... 34
4.5 Área foliar..... 34
4.6 Análise estatística..... 35
5 RESULTADOS E DISCUSSÃO..... 36
5.1 Manejos de irrigação automática..... 36
5.1.1 Manejo de irrigação automática via solo..... 36
5.1.2 Manejo Automático da irrigação via clima..... 37
5.1.3 Manejo da irrigação manual via solo..... 39
5.1.4 Manejo de irrigação manual via clima..... 40
5.2 Variáveis de crescimento..... 42

5.3 Variáveis de produção..... 46
5.4 Eficiência do uso da água..... 49
5.5 Análise econômica..... 51
5.6 Comparação com sistemas existentes..... 52
6 CONCLUSÕES..... 54
REFERÊNCIAS..... 55
ANEXO A – CODIGO FONTE COMENTADO DO SISTEMA DE IRRIGAÇÃO AUTÔNOMO VIA SOLO 61
ANEXO B – CODIGO FONTE COMENTADO DO SISTEMA DE IRRIGAÇÃO AUTÔNOMO VIA CLIMA 82

1. INTRODUÇÃO

A escassez hídrica e o aumento da tarifação da água e energia afetam diretamente a agricultura brasileira, os novos rumos culturais, fazem a necessidade de produtos mais saudáveis, de melhor qualidade e com menos agrotóxicos. A busca pelo menor custo com mão de obra, fazem com que os processos da agricultura estejam se modernizando de diferentes maneiras. Uma das principais atividades a sofrerem modernização são os métodos de irrigação, que passam cada dia mais a serem automatizados. Dentre várias formas de atualizações da irrigação, estão inclusos o uso de sensores de umidade de variadas metodologias, ligados a computadores e controladores, que por meio de sinais digitais, interligam o sistema de irrigação, controlando todo o manejo da irrigação e aplicação de nutrientes (MELO JÚNIOR, 2013). Outra forma de automação é com a utilização de programas computacionais, que se comunicam de forma remota com os equipamentos, como por exemplo, sistemas de irrigação por pivô central, onde todo o conjunto embora trabalhe de forma independente, combina-se em um funcionamento contínuo e organizado para manter uma estrutura de irrigação linear. Esse processo de inovação visa reduzir os custos com energia elétrica, água e mão-de-obra.

Embora seja uma ótima solução para baratear os processos e ter um controle maior sobre o plantio, a inserção de sistemas automatizados continua sendo um desafio. Isso porque para realização de tal processo, é preciso um investimento inicial alto e mão de obra especializada, assim, muitos produtores de menor porte acabam ficando excluídos desses métodos (SOARES FILHO et al., 2015). Para amenizar essa situação, a utilização de ferramentas de menores custos e sistemas que embora mais simples façam o suficiente para realizar as funções com exatidão, podem levar a modernização aos pequenos produtores.

A robótica, a inteligência artificial e a aprendizagem de máquina já estão no cotidiano da sociedade, desde a educação até a agricultura, portanto é totalmente plausível inserir na pequena produção rural unidades automatizadas (MATARIC, 2014). É possível obter tecnologias avançadas para agricultura familiar com baixo custo, inclusive podem reduzir a mão-de-obra para produtores deste setor. O principal desafio para a robótica na agricultura é o equilíbrio entre diversos fatores, são eles: custo, tamanho, alimentação, desempenho e complexidade com as funções requeridas (HACKENHAAR, HACKENHAAR e ABREU, 2015).

A tecnologia vem mudando o rumo da sociedade e também da agricultura. Propostas e inovações surgem todos os dias, e no cenário atual, inovações tecnológicas para a agricultura familiar são importantes para o aumento da produção de pequenos

produtores, que possuem baixo capital de investimento.

2. HIPÓTESE E OBJETIVOS

2.1. Hipótese

A inserção de um sistema autônomo de baixo custo controlado via aplicativo móvel no manejo da irrigação, que facilita o cultivo, diminui a mão de obra e melhora a produção de hortaliças.

2.2. Objetivo geral

Desenvolver e avaliar o desempenho de um protótipo de um sistema autônomo de baixo custo para manejo da irrigação na produção de hortaliças, gerando economia de energia e mão de obra.

2.3. Objetivos específicos

- Desenvolver um sistema automatizado para manejo da irrigação com uso de dados climáticos e de baixo custo acionado via aplicativo móvel.
- Desenvolver um sistema automatizado para manejo da irrigação com uso de dados via solo, acionado via aplicativo móvel.
- Analisar a viabilidade dos sistemas no ambiente agrícola.
- Analisar o impacto na mão de obra aplicada e economia de recursos.

3. REVISÃO DE LITERATURA

3.1. Tecnologia no desenvolvimento agrícola: agricultura digital

Constantemente é observado a necessidade de aumento da eficiência em todos os setores da economia, isso para que continue a existir competitividade. Na agricultura a linha de pensamento é a mesma. Com o avanço da informática e das diversas tecnologias de processamento de dados e automação de processos, cada dia que se passa, o produtor rural está se tornando um empresário, por controlar cada vez mais a linha de produção do ambiente agrícola. Dentre os desafios desta nova geração de produtores é o manuseio de propriedades heterogêneas onde cada parte da propriedade precisa de uma atenção particular para a tomada de decisão. Cada parte de uma propriedade tem suas próprias características e quando tratadas assim, pode-se melhorar os rendimentos e aproveitar as diferenças dentre elas e tornar isso uma vantagem a ser aproveitada. É nesta estratégia que entra a agricultura digital, onde softwares e dispositivos eletrônicos ajudam a coletar e processar informações e dados para uma melhor aplicação de insumos, vinculando esse processamento de dados com aplicações como a irrigação de precisão, que se pode aumentar a produção de cultivo.

A irrigação de precisão é um termo aplicado ao manejo da irrigação com o auxílio de ferramentas como sensoriamento remoto, automação de sistemas de irrigação com uso de sensores (SILVA et al., 2020; MONTELEONE et al., 2020), uso de sistemas de informação geográfica (OLDONI; BASSOI, 2016) e etc., que visa a aplicação da lâmina de irrigação necessária (MANTOVANI et al., 2009), no tempo e local correto para as necessidades da cultura, gerando assim economia de recursos. Esse tipo de irrigação faz parte da agricultura de precisão (TSCHIEDEL; FERREIRA, 2002; SILVA et al., 2020) e torna-se a fundamental para uma agricultura a cada dia mais tecnificada. A agricultura digital unida a de precisão consegue melhorar a aplicação de insumos agrícolas e o gerenciamento dos recursos alcançando um aumento na produtividade das lavouras (TSCHIEDEL; FERREIRA, 2002). Controlar e agrupar informações agronômicas e atender as necessidades de partes do campo e não de necessidades para campos inteiro, que também é uma meta desse tipo de metodologia, dividindo assim, campos inteiros em zonas menores e homogêneas, e aplicando nestes, os insumos necessários. (TSCHIEDEL et al., 2002).

A agricultura de precisão (MOLIN; AMARAL; COLAÇO, 2015) engloba o uso de tecnologias atuais para o manejo de solo (OLDONI et al., 2019), insumos e culturas (ROCHA et al., 2019), de modo a adequar às variações espaciais e temporais em fatores que afetam a produtividade das plantas (TSCHIEDEL et al, 2002). Uma metodologia da agricultura de

precisão é a utilização de sensores (MONTELEONE et al., 2020) e sistemas robóticos, capazes de captar características do cultivo e atuar de acordo com suas necessidades, atendendo as demandas hídricas da cultura de maneira imediata e eficiente.

3.2. Automação na agricultura irrigada

Sistemas autônomos para irrigação já são reais no nosso país, mas normalmente utilizado de formas básicas, como iniciar uma irrigação em um determinado horário e encerrar em outro, ações diferentes dessas ainda são pouco aplicadas no Brasil, principalmente no âmbito da população com menos recursos, podendo-se atribuir este fato a diversos fatores como o baixo interesse pelos profissionais da área no uso da automação, custos envolvidos que geralmente são elevados e a complexidade que envolve o processo, pois a agricultura é dinâmica, principalmente a irrigação, na qual há diversas variáveis como clima, solo e planta (MANTOVANI et al., 2009) envolvidas de maneira conjunta.

O uso de unidades robóticas na agricultura mundial está normalmente relacionado a processos como colheitas, preparo do solo, obtenção de amostras de solo ou plantas e também analise de produção com uso de aeronaves não tripuladas, os famosos drones, também são utilizadas imagens por satélite (CALOU et al., 2020; ROCHA NETO et al., 2017). A irrigação, parece ter ficado afastada dessa tecnologia, sendo esta utilizada apenas em sistemas automatizados a base de microcontroladores para simples tarefas. Contudo, estes sistemas normalmente se baseiam apenas em um fator para tomadas de decisão como o tempo de irrigação. Batista et al. (2017) em estudos sobre uma unidade autônoma para irrigação em comparação a sistemas convencionais, demonstraram que a eficiência desses sistemas para irrigação pode ser tão boa quanto os sistemas convencionais.

Contudo, existem diversas maneiras de aumentar a eficiência da irrigação, seja por imagens de satélite, monitoramento eficiente da umidade do solo ou medidas de evapotranspiração em tempo real. O gerenciamento de irrigação e nutrientes pode ser realizados através do uso de técnicas de sensoriamento, onde usando dados de sensores, com modelos de estado ideais é possível realizar a gestão eficiente do uso da água. De posse dos dados é possível analisar se é necessário ou não, usar diferentes estratégias de irrigação em diferentes áreas de cultivo, devido a variabilidade natural do solo e do uso da água pelas plantas, assim gerando economia de recursos (SILVA et al., 2020).

O uso de sistemas autônomos na irrigação, além de toda a estrutura dinâmica necessária, como atuadores para aplicação de água, o uso de sensores que permitam acesso de

informações sobre o solo, planta e temperatura são fundamentais para que a própria unidade possa tomar a decisão do tempo de rega necessária com base nessas informações, além de promover irrigações em períodos essenciais para cultura durante o dia, através de pulsos, ou seja, pequenas lâminas proporcionadas ao longo do dia.

Tão importante como a atuação do sistema também é a comunicação com os sensores e também outros métodos de controle, afim de manter uma rede de informações concisas e existir uma relação de segurança quanto aos dados. Essa comunicação pode ser via cabos ou mesmo sistemas sem fio, como rádio frequência (DOTA et al., 2010). Atualmente aplicativos instalados em celulares estão promovendo a tomada de decisão em sistemas de irrigação, onde os sistemas podem ser acionados de qualquer lugar apenas com acionamento por parte do produtor (SOUZA et al., 2019b). Tal fato, além de promover praticidade pode promover maior eficiência, já que sinais sobre o momento de irrigar podem ser obtidos a qualquer momento, sem a necessidade de uma visita a área de produção.

3.3. Utilização de sensores na agricultura

O uso de sensores é uma realidade na maioria dos países que utilizam agricultura de precisão. Na irrigação, o uso de informações obtidas através destes sensores pode auxiliar nas estratégias a serem executadas, para que eles apliquem de maneira diferenciada a quantidade de água necessária para as culturas (SILVA et al., 2020).

Os sensores mais usados na agricultura são os de umidade do solo, como as sondas TDR e FDR (SOUZA et al., 2013) e tensiômetros (LIBARDI, 2012). Os tensiômetros são capazes de identificar a umidade com base na pressão exercida pela movimentação de água no solo, onde o manejo da irrigação é obtido através de uma curva característica relacionando as umidades da capacidade de campo com a umidade atual do solo. Diversos trabalhos já comprovaram a eficiência do uso de tensiômetros (FELICIO et al., 2019; SILVA et al., 2015), contudo estes sistemas requerem observações muitas vezes diárias, necessitando assim de mão de obra atuante na área de produção.

Outro modelo de sensor de umidade é o higrômetro de imersão, que é composto basicamente por dois eletrodos que ficaram enterrados e conduzem uma corrente, assim o resultado é a condutibilidade do solo onde o sensor está. Quando o solo estiver seco, a resistência entre os eletrodos do sensor de umidade de solo irá aumentar dificultando a passagem de corrente e o contrário também acontece, quando mais água tiver no solo, maior unidade, então a resistência entre os eletrodos diminuirá permitindo a passagem de corrente

(CARVALHO e OLIVEIRA, 2009; KLAR, 1988). Através da resistência entre esses dois eletrodos poderemos descobrir se o solo está muito úmido ou muito seco por comparação, esse aparelho já vem calibrado e com uma tabela de referência de umidade comparada com a resposta do sensor, dados que são usados no sistema robótico.

3.4. Cultivo em ambiente protegido

O ambiente protegido é responsável por várias alterações nos diversos elementos meteorológicos, tornando viável a produção de vegetais em épocas ou locais cujas condições climáticas são críticas (AGUIAR et al., 2002). Em regiões onde a baixa temperatura é fator limitante, o uso de estufas é uma alternativa eficiente para obter uma maior regularidade de produção de culturas como a alface nos meses de inverno (SEGOVIA et al., 1997).

O cultivo do pimentão em ambiente protegido, por exemplo, possibilita uma produção contínua e certa, abastecendo o mercado o ano todo. Nestas condições são obtidos melhores desempenhos vegetativos e, consequentemente, melhor produtividade e qualidade dos frutos, em épocas do ano em que as condições ambientais são desfavoráveis no campo. Em cultivo protegido, o comportamento da cultura e sua necessidade nutricional são diferentes (SGANZERLA, 1995; ARAGÃO et al., 2011) onde as temperaturas ambientes são diferenciadas e a fertirrigação é comumente utilizada.

Diversos avanços tecnológicos para a agricultura são comumente utilizados nos cultivos em ambientes protegidos, dentre eles pode-se destacar os sistemas hidropônicos, o uso da automação por meio de sensores, uso de iluminação a base de lâmpadas do tipo led, dentre outras tecnologias. Segundo Purquerio e Tivelli (2009) a produtividade em ambiente protegido supera a de campo em até três vezes e com qualidade superior, fundamentalmente isso é possível devido ao rigoroso controle utilizado nas casas de vegetação, reduzindo assim, a incidência de pragas e aumentando a eficiência dos insumos. Contudo é importante ressaltar o elevado custo investido, sendo apenas possível atingir a lucratividade com culturas de alto valor agregado.

3.5. Método de controle IoT

Internet das Coisas (do inglês *internet of things* ou IoT) vem crescendo a cada dia mais, principalmente com o avanço dos sistemas embarcados, microcontroladores, sensores, entre outros. De acordo com Santos et. al. (2016), a internet das coisas “nada mais é que uma extensão da internet atual, que proporciona aos objetos do dia a dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à internet”.

Tendo o conhecimento do que se trata, fica mais fácil entender seu poder, com essa capacidade de comunicação é possível comunicar sistemas, componentes, equipamentos e fazer com que eles trabalhem juntos para desenvolver um objetivo. No caso desse projeto, esse objetivo seria o benefício do manejo da irrigação com aplicação mais precisa, com a ajuda de sensores que podem ter seus dados transferidos para um controlador que tome decisões de acionamento e entreguem tudo isso ao usuário do sistema de forma remota e em tempo real (SILVA et al., 2020). Tudo isso atrelado a um controle sobre as ocorrências como a ação da temperatura e umidade, criando alertas para o operador do sistema em momentos críticos, evitando assim a necessidade de uma presença humana no local em tempo integral.

A internet das coisas se tornou uma das tecnologias mais importantes do século XXI. Sendo possível conectar objetos do dia a dia: eletrodomésticos, carros, sensores, alarmes, à internet por meio de dispositivos incorporados, é possível uma comunicação perfeita entre pessoas, processos e objetos (MATARIC, 2014).

Tudo isso graças a incorporação de sistemas como a computação de baixo custo, nuvem1, big data2, análise e tecnologias móveis, é possível compartilhar e coletar dados com a mínima intervenção humana, onde o mundo físico encontra o digital, por meio da hiperconectividade, fica mais fácil que sistemas digitais possam monitorar, registrar e tomar decisões interagir com o mundo (HACKENHAAR, HACKENHAAR e ABREU, 2015).

Para que a IoT pudesse funcionar e tornar seu uso prático, foi necessária a atuação, aplicação e desenvolvimento de outras tecnologias, dentre elas: tecnologia de sensores de baixo custo e baixa potência, acessíveis e confiáveis; o aumento das possibilidades de conexão e protocolos de comunicação; aumento de plataformas de processamento em nuvem; *machine learning* e análise avançada; redes neurais, que trouxeram o processamento de linguagem natural para dispositivos IoT e os tornaram atraentes, acessíveis e viáveis para uso doméstico (SANTOS et al., 2016).

Com tantas opções de uso e funcionalidades e facilidades, o uso da IoT na agricultura é uma opção promissora para melhorar o trabalho no campo e utilizar-se da tecnologia para ajudar no desenvolvimento e melhoramento de custos e de produção.

[1] Armazenamento de dados em sistema de servidores online.
[2] Análise e a interpretação de grandes volumes de dados de grande variedade.

4. MATERIAIS E MÉTODOS

4.1. Caracterização experimental

O ambiente utilizado para a aplicação, testes e experimento foi uma casa de vegetação do Departamento de Engenharia Agrícola – DENA, localizada na estação meteorológica da Universidade Federal do Ceará – UFC, campus do PICI (Figura 1) nas coordenadas geográficas 3°44'45"S, 38°34'55"W e 19,5 m de altitude acima do mar.

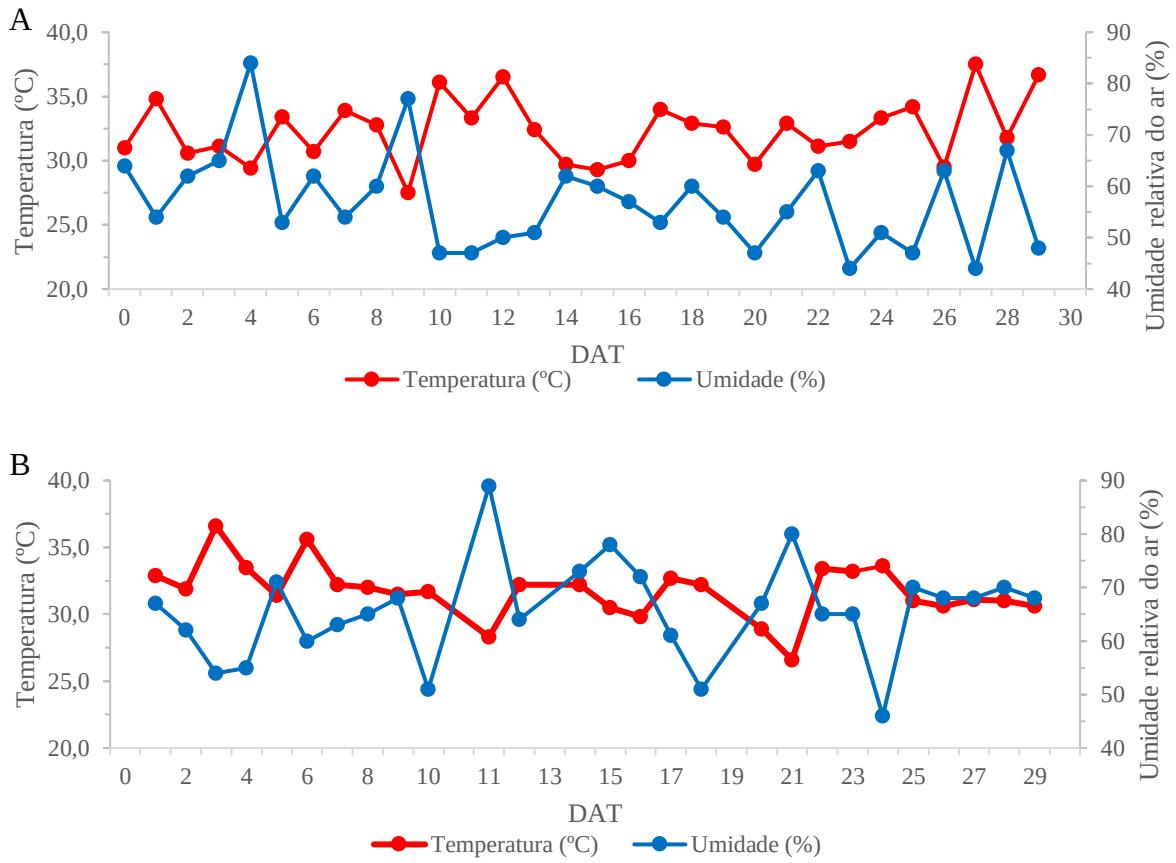
Figura 1 - Localização da casa de vegetação

Fonte: adaptado de www.google.com.br

A casa de vegetação possui dimensões de 6,25 metros de largura e 12 metros de comprimento, uma cobertura em arco simples com filme de polietileno de baixa densidade com 0,10 mm de espessura e uma tela de sombreamento (50%) em seu interior na cor preta. Durante o ciclo, foi monitorada a temperatura ambiente (°C), a umidade relativa do ar (%) e a evaporação da água (mm dia^{-1}) conforme a Figura 2.

A temperatura ambiente apresentou média de 32,2 °C enquanto a umidade relativa do ar observada foi de 54% durante o ciclo 1 de produção (Figura 2A) que foi realizado entre os dias 17 de dezembro de 2020 a 16 de janeiro de 2021. No ciclo 2 (Figura 2B) ocorrido entre 1 de fevereiro a 3 de março de 2021, a temperatura ambiente apresentou média de 29,6 °C enquanto a umidade relativa do ar observada foi de 65%. A evaporação da água no tanque classe "A" ficou em média com valores de 4,34 mm dia^{-1} para o ciclo 1 e 2,9 mm dia^{-1} para o ciclo 2.

Figura 2 - Dados climáticos no interior do ambiente protegido: temperatura e umidade do ar para o ciclo 1 (A) e para o ciclo 2 (B)



Fonte: dados da pesquisa

O solo utilizado no experimento foi classificado como um Argisolo Vermelho-Amarelo (SANTOS et. al., 2013), possuindo as seguintes características físicas: areia: 62 %; silte: 10 % argila: 28 % e densidade do solo: 1,52 g cm^{-3}. As características químicas podem ser observadas na Tabela 1.

Tabela 1 - Características química de solo utilizado no experimento

Complexo Sortivo ($cmol_c$ kg^{-1})							
Ca^{2+}	Mg^{2+}	Na^{+}	K^{+}	$H^{+} + Al^{3+}$	Al^{3+}	S	CTC
1,20	0,60	0,23	0,36	1,98	0,15	2,6	4,37
pH em água	P (mg kg^{-1})	CE (dS m^{-1})	C (g kg^{-1})	N (g kg^{-1})	M O (g kg^{-1})		
6,0	32	0,35	6,48	0,61	11,17		

Fonte: dados da pesquisa

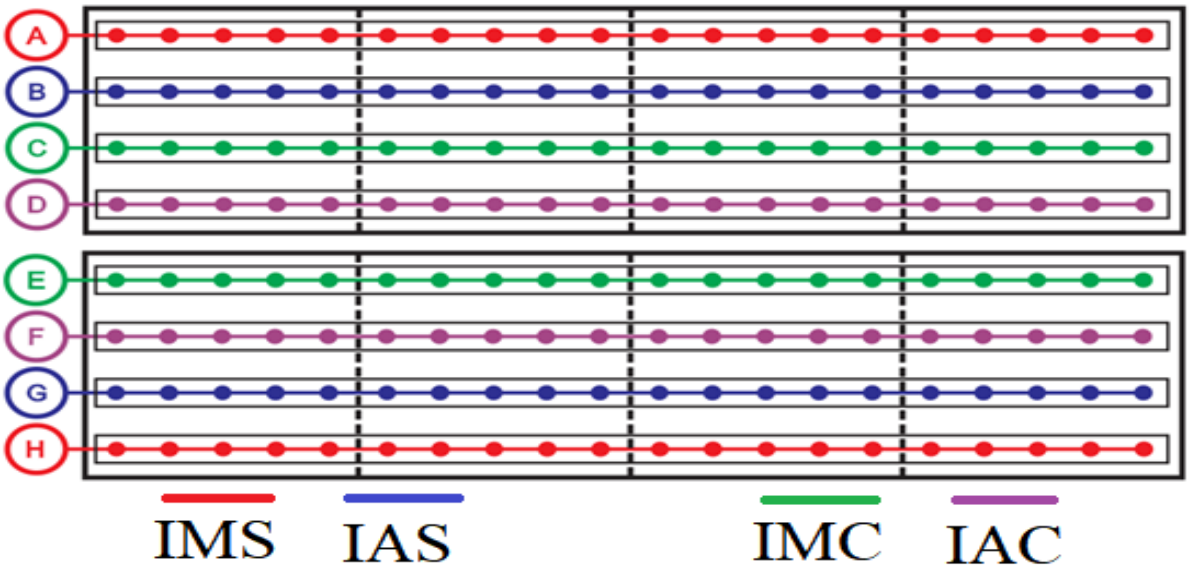
4.2. Cultura utilizada no experimento

A cultura utilizada foi a rúcula (*Eruca Sativa* L.), semeada em bandejas de poliestireno 200 células com substrato de pó de coco e vermiculita. Foram semeadas duas sementes por célula e após aparecimento da terceira folha, foram escolhidas as mais vigorosas e realizado assim, o desbaste de plantas com alguma deficiência ou baixo vigor. Aos 30 dias após a semeadura, as plantas foram transplantadas para as unidades experimentais e irrigadas com lâminas constantes para garantir um bom desenvolvimento, tentando amenizar o *stress* gerado pelo transplantio. A adubação foi realizada via fertirrigação com as doses de 40 kg ha^{-1} de N, 300 kg ha^{-1} de P e 100 kg ha^{-1} de K_2O, conforme recomendação de adubação de Trani (1997), para tanto foram utilizados os seguintes fertilizantes: Ureia (45% de N), MAP (12% de N e 61% de P_2O_5) e Dripsol NKS (45% de K_2O e 12% de N). Foram aplicados 50% dos nutrientes aos 10 dias após o transplanio (DAT) e 50% aos 20 DAT, onde foram diluidos em água e inserido no solo via fertirrigação. Foram conduzidos dois ciclos de produção, o ciclo 1 compreendido entre 17 de dezembro de 2020 à 16 de janeiro de 2021; o ciclo 2 entre 01 de fevereiro de 2021 à 03 de março de 2021.

4.3. Tratamentos e delineamento experimental

O experimento foi realizado em blocos casualizados com 4 tipos de manejo, dividido em sistemas automáticos e manuais: irrigação automática via solo (IAS); irrigação automática via clima (IAC), irrigação manual via solo (IMS) e irrigação manual via clima (IMC) conforme Figura 3. A parcela experimental foi constituída por canteiro com dimensões de 1,5 m x 0,30 m, possuindo 5 plantas espaçadas em 0,30 m. Cada tratamento possuía 8 repetições, totalizando assim, 32 parcelas experimentais.

Figura 3 - Visão geral do delineamento experimental e disposição dos tratamentos



Fonte: elaborado pelo autor

4.3.1. Manejo de irrigação manual via solo

O manejo IMS foi realizado por tensiômetros de punção com medição realizadas por tensímetro digital e com o auxílio da curva característica (Figura 4) para determinação da umidade do solo e posteriormente foi determinada a irrigação real necessária a ser aplicada (Eq.1).

$$IRN = (\theta_{cc} - \theta_{atual}) x\ Z\ \text{x KL} \quad (1)$$

Em que:

IRN – Irrigação real necessária (mm);

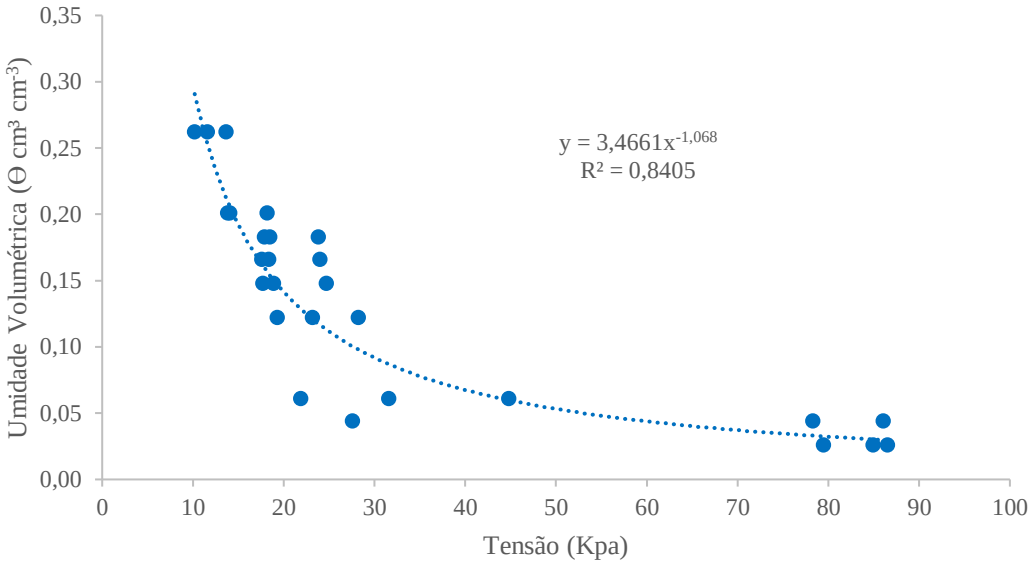
Θ_{cc} – Umidade em capacidade de campo (cm^{-3} cm^{-3});

Θ_{atual} – Umidade atual do solo (cm^{-3} cm^{-3});

Z – Profundidade do sistema radícular da cultura (mm);

KL – Coeficiente de localização (BERNARDO et al., 2019).

Figura 4 - Curva característica do solo obtida para realização do manejo da irrigação



Fonte: dados do solo

No IMS a aplicação foi realizada manualmente, com a multiplicação da lâmina pela área da parcela experimental, utilizando um recipiente graduado, sendo realizada uma vez por dia. Para estabelecimento inicial da cultura foi estabelecida uma lâmina constante. A partir de cinco dias após o transplantio (DAT), foram obtidas as umidades e calculado o volume a ser aplicado na unidade experimental.

4.3.2. Manejo de irrigação manual via clima

O manejo IMC foi realizado através da evapotranspiração da cultura, obtida pela determinação da evapotranspiração de referência pelo tanque classe A (Figura 5), através de medidas de evaporação do tanque (ECA), coeficiente do tanque e o coeficiente da cultura, conforme Allen et al., (1998). No IMC a aplicação foi realizada manualmente uma vez ao dia, com a multiplicação da lâmina (ETc) pela área da parcela experimental, utilizando recipiente graduado. Utilizando todos esses dados, chegamos a seguinte Equação 2:

$$IRN = ECA\ x\ Kp\ x\ Kc\ x\ KL \quad (2)$$

Em que:

IRN – Irrigação real necessária (mm);

ECA – Evaporação do tanque classe “A” (mm dia^{-1})

Kp – Coeficiente do tanque

Kc – Coeficiente da cultura

KL - coeficiente da irrigação localizada (BERNARDO et al., 2019).

O coeficiente de cultivo da cultura da rúcula utilizado, teve como base as informações obtidas por Villares et al. (2011) com valores de 0,29 (0-8 DAT), 0,52 (9-16 DAT), 0,93 (17-24 DAT), 0,87 (24-30 DAT).

Figura 5 - Tanque classe A utilizado no experimento

4.3.3. Manejo da irrigação automática via solo

O manejo IAS, consiste em um dispositivo que desempenha a função de irrigador com análise baseada em sensores de umidade do solo e capazes de identificar a umidade baseada no coeficiente elétrico do solo, pois quanto mais úmido, maior sua condutibilidade.

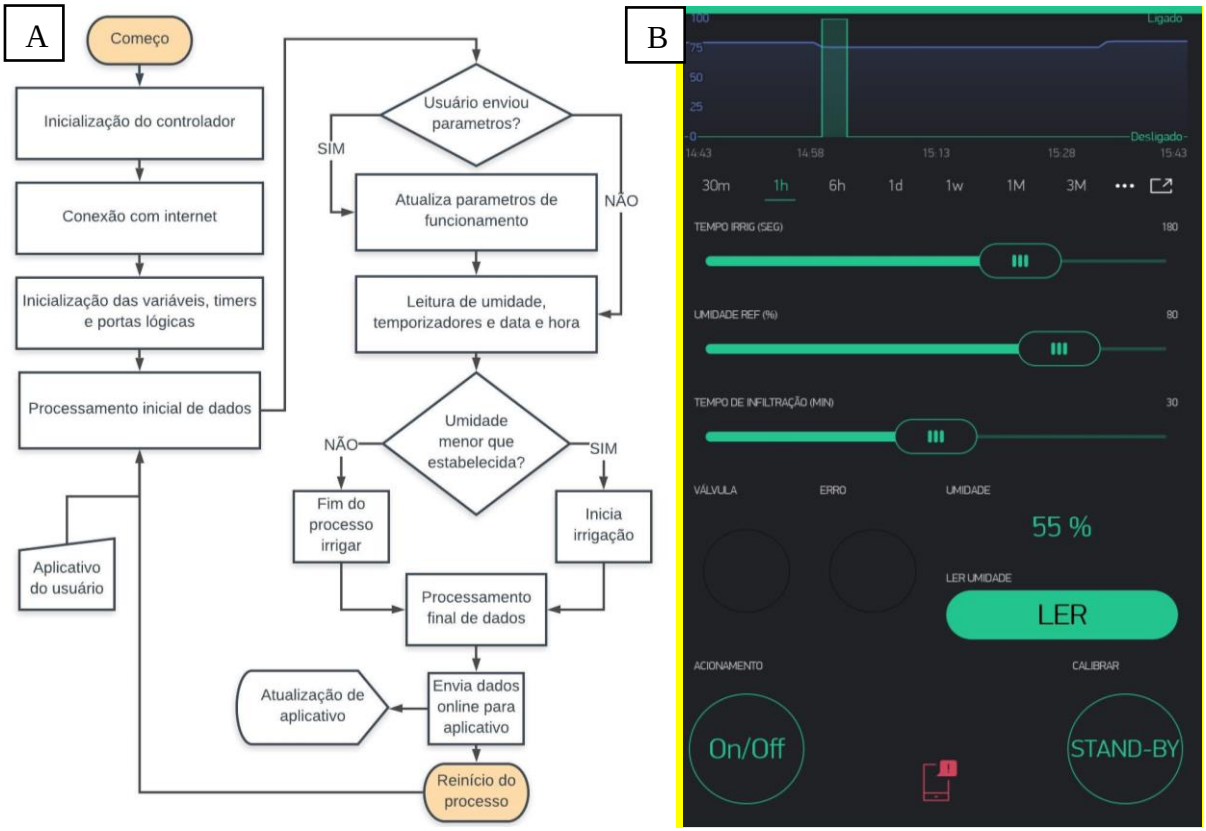
A estrutura do programa se divide em acionamentos de válvulas, leitura de umidade do solo e salvamento dos dados para criação de um *datalogger*, além de transmitir esses dados em tempo real para o usuário através de um aplicativo para *smartphone*, de modo que o mesmo possa ter conhecimento da situação da cultura. O passo a passo do funcionamento pode ser observado na Figura 6A, que representa o fluxograma da execução do programa.

A cada ciclo de processo o sistema atualiza a interface do usuário para que tenha

uma informação em tempo real no próprio aplicativo, no qual, existe a opção de acionamentos tanto manual como automático, utilizado de acordo com a necessidade do usuário, como pode ser observado na Figura 6B. Na opção manual o usuário tem o controle de informações em tempo real dos valores de umidade das unidades experimentais e pode controlar o acionamento da irrigação, além de fazer o sistema realizar leituras para identificar a situação atual de umidade do solo.

No modo automático o usuário deverá definir o parâmetro para acionamento do sistema, no qual é a umidade do solo desejada. Daí então o sistema entrará em ação e fará o possível para manter as condições pré-estabelecidas. Sendo necessário o usuário ainda definir qual o tempo base de irrigação (0 a 180 segundos) e definir o tempo de infiltração (0 a 60 minutos). A partir daí em com tempos pré-definidos o sistema realizará novas leituras e disponibilizar na tela do aplicativo, e quando necessário, atuará realizando o manejo da irrigação. Uma das grandes vantagens desse sistema e a capacidade armazenar cada valor de variável, minuto a minuto, no servidor do programa e mantê-los armazenados.

Figura 6 - Organograma de funcionamento do dispositivo IAS (A) e interface de comando no aplicativo do dispositivo IAS (B).

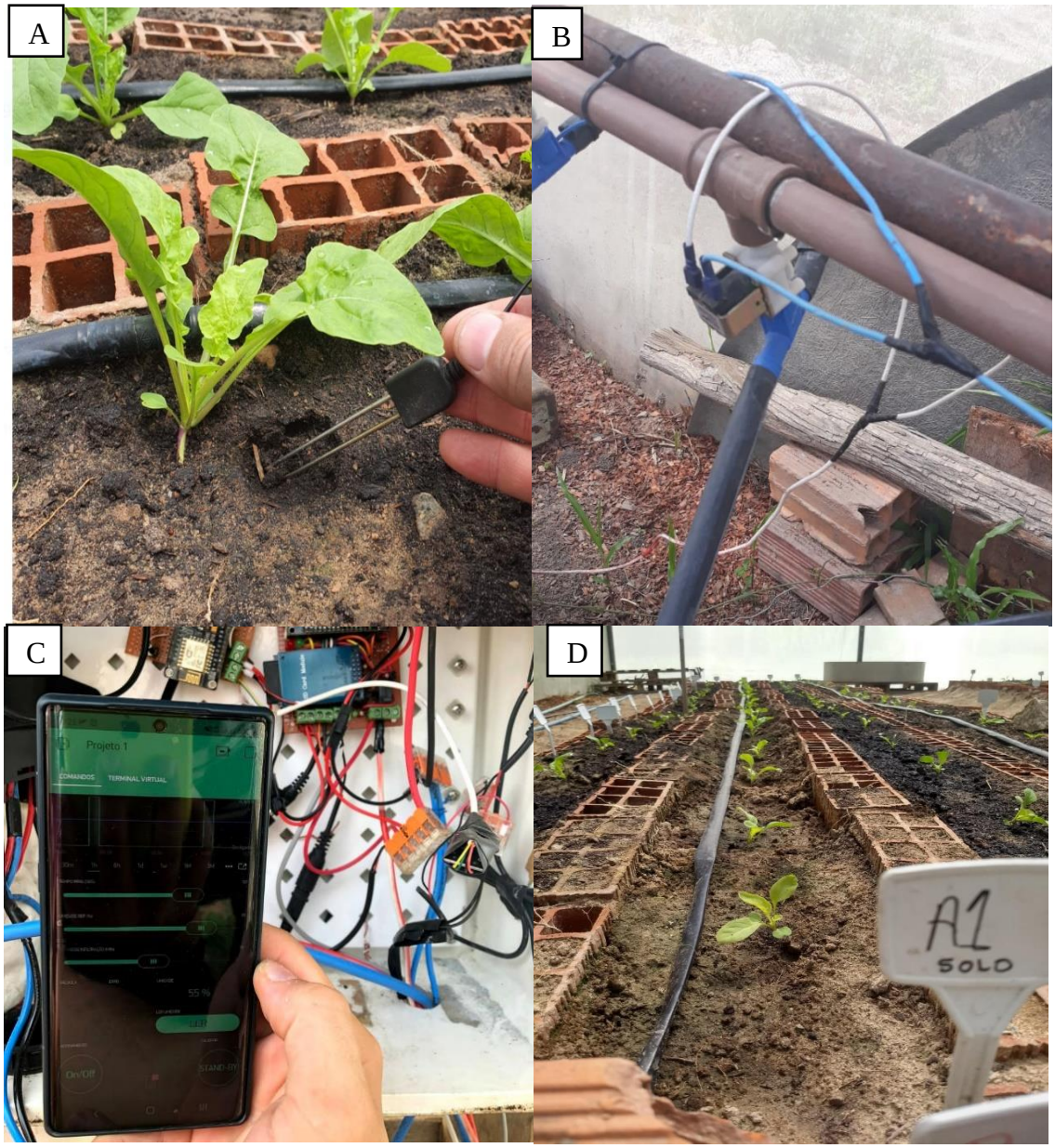


Fonte: elaborado pelo autor

Para o IAS foi utilizado o sensor imersivo do tipo eletrodo, com material metálico resistente a corrosão, com a capacidade de variação de resistência de acordo com a umidade do solo (Figura 7A). Os dados armazenados na memória do microcontrolador instalado no ambiente protegido são enviados para um servidor online acessível pelo aplicativo de celular (Figura 7B). Após obtidos os dados, são realizados os comandos da irrigação através de válvulas solenoides (Figura 7C) que faziam a liberação da água proveniente de uma caixa d'agua instalada a 2 m de altura e no local, para a tubulação de irrigação. A partir de então a irrigação foi realizada por fita de gotejamento com vazão de 0,68 $L\,h^{-1}$ (Figura 7D) por emissor. As vazões dos emissores foram medidas durante a realização do experimento.

O sensor de umidade pode ser calibrado de acordo com o solo utilizado, de forma a se adaptar. Essa calibração ocorre com leituras das tensões em 2 situações, em estado seco do solo e com o solo completamente úmido. O intuito é identificar as tensões de resposta do sensor em ambas as situações e gerar limitadores de tensão máxima e mínima de atuação. Assim, o sistema identifica os limites e define as tensões proporcionais entre esses valores, podendo-se definir uma porcentagem baseada nesses limites. Outra forma de calibrar o sensor é retirando esses limites e liberando toda a capacidade de leitura do sensor. Então faz-se a inserção de água nas proximidades do sensor até que esteja na situação considerada ideal para a cultura (pode ser utilizado um tensiômetro, por exemplo, para identificar esse ponto), pelo aplicativo, faz-se a leitura do sensor e baseado na resposta do mesmo, define-se aquele ponto como a porcentagem definida para acionamento. Assim, quando a tensão de água no solo for menor que a definida, o sistema automaticamente realizará a inserção de água no solo, por via das fitas gotejadoras.

Figura 7 - Sensor de umidade utilizado e armazenamento de dados (A), válvula solenóide instalada (B), aplicativo móvel (C) e fita gotejadora (D)



Fonte: elaborado pelo autor

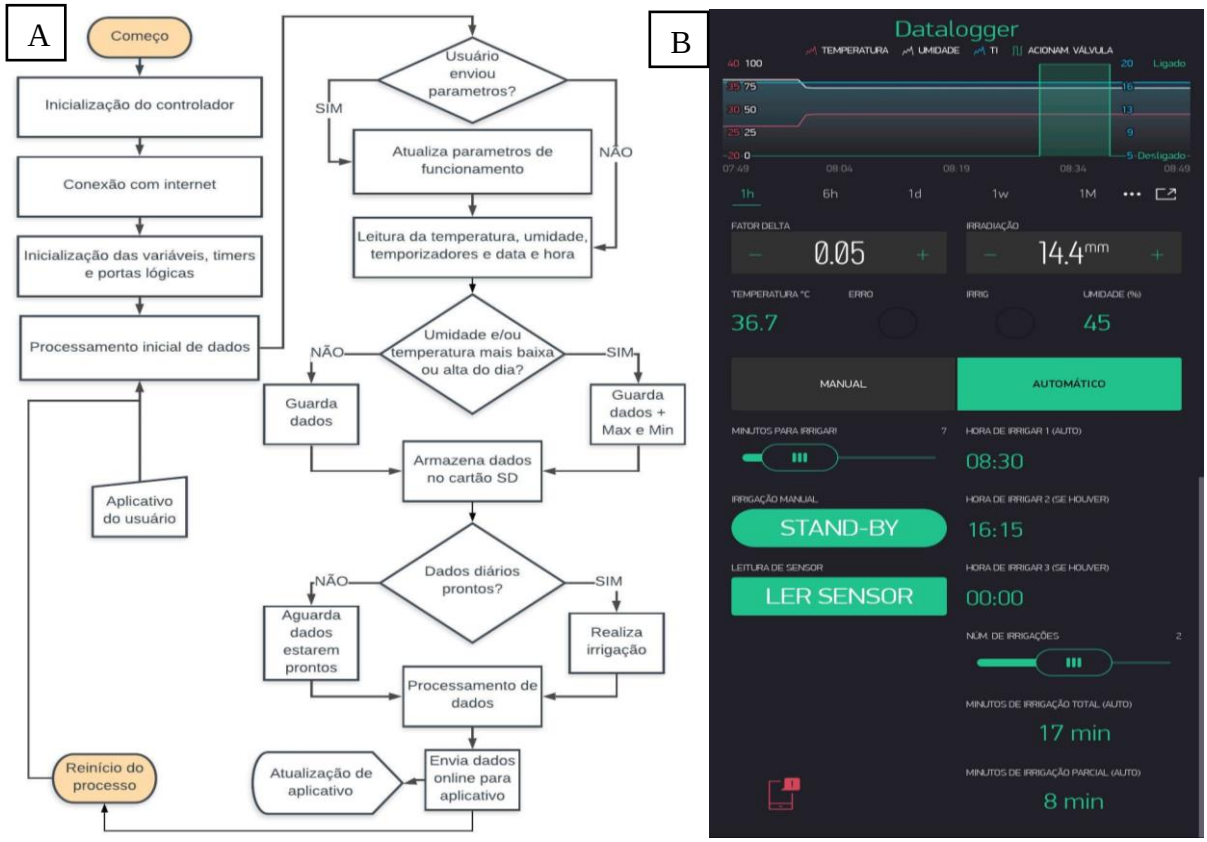
4.3.4. Manejo da irrigação automática via clima

O manejo IAC consistiu em um dispositivo que realiza a aplicação de água no solo, mas com a capacidade de um sistema autônomo que capta a temperatura e umidade do ambiente decorrente do dia e armazena em um banco de dados.

Com a obtenção da radiação solar, obtida através do balanço de energia (DOORENBOS e PRUITT, 1977), o controlador realiza o cálculo da evapotranspiração de referência por Hargreaves-Samani (ALLEN et al., 1998) e consequentemente a lâmina de água

que deve ser aplicada nas unidades experimentais com o uso do Kc. O passo a passo do funcionamento pode ser observado na Figura 8A, que representa o fluxograma da execução do programa. A interface de uso do usuário pode ser vista na Figura 8B.

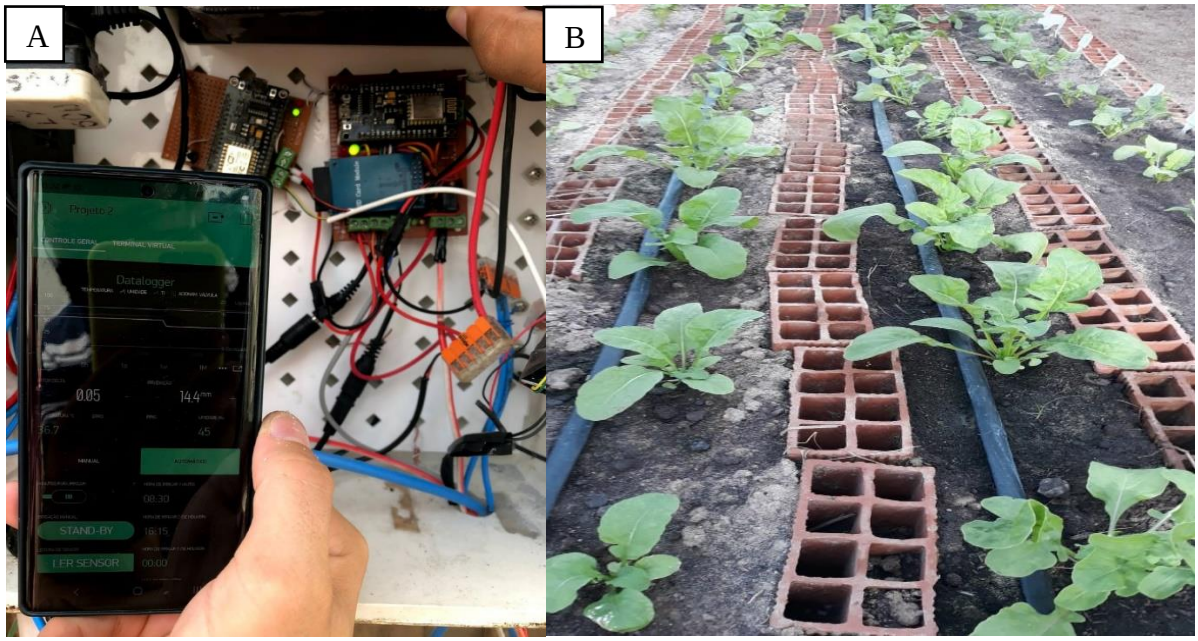
Figura 8 - Organograma de funcionamento do dispositivo IAC (A) e design de interface do aplicativo.



Fonte: elaborado pelo autor

O aplicativo foi desenvolvido e utilizado através de smartphone (Figura 9A), onde, de posse dos dados, acionava-se o sistema de irrigação através de válvula solenóide e com o sistema de gotejamento, aplicava-se a lâmina de irrigação necessária (Figura 9B). Para tanto considerou-se uma eficiência de irrigação de 90% (BERNARDO et al., 2019).

Figura 9 - Componentes do manejo autônomo de irrigação via clima utilizado no experimento



Fonte: elaborado pelo autor

A utilização é intuitiva, pois o usuário tem a opção de uso manual ou automático, onde no modo manual, pode-se realizar a leitura dos sensores e realizar uma irrigação temporizada. Já no modo automático o usuário deve inserir o número de irrigações diárias desejada e os horários, ainda precisa ser informado um "fator delta" que é uma sintetização de variáveis da cultura. Os dados do fator delta são: coeficiente da cultura, coeficiente da irrigação localizada, área da planta, eficiência do sistema de irrigação, número de emissores por planta, vazão do gotejador.

Para o cálculo de fator delta da equação utilizada pelo controlador para a irrigação, faz-se necessário o uso do método de Hargreaves-Samani (equação 3) e equações de cálculo de tempo de irrigação a seguir.

$$ETo = 0{,}0023\ x\ Ra\ x\ \sqrt{(Tmáx - Tmín)}\ x\ (Tméd + 17{,}8) \quad (3)$$

$$ETc = ETo\ x\ Kc \quad (4)$$

$$ETc_{loc} = ETc\ x\ KL \quad (5)$$

$$ITN = \frac{ETc_{loc}}{Ea} \quad (6)$$

$$Ti = \frac{(ITN\ x\ A)}{n\ x\ q} \quad (7)$$

Onde simplificando em uma única equação, temos:

$$Ti = 0{,}0023\ x\ \sqrt{(Tmáx - Tmín)}\ x\ (Tméd + 17{,}8)\ x\ Ra\ x\ \frac{Kc\ x\ KL\ x\ A}{Ea\ x\ n\ x\ q} \quad (8)$$

Assim, temos que:

$$Fator\ \Delta = \frac{Kc\ x\ KL\ x\ A}{Ea\ x\ n\ x\ q} \quad (9)$$

Onde,

ETo - evapotranspiração basica, (mm.dia^{-1});

ETc - evapotranspiração da cultura, (mm.dia^{-1});

Kc - coeficiente da cultura;

ETc_{Loc} - evapotranspiração localizada, (mm.dia^{-1});

ITN - irrigação total necessária, (mm.dia^{-1});

Ti - tempo de irrigação, em horas;

KL - coeficiente da irrigação localizada (BERNARDO et al., 2019);

Ea - rendimento da irrigação utilizada (0,9);

A - área referente a cultura na unidade experimental, (m²);

N - número de irrigadores por planta;

q - vazão do gotejador, (L h^{-1});

Ra - radiação solar no interior da casa de vegetação, (mm.dia^{-1}).

O fator delta foi desenvolvido pensando nos valores limitados que podem ser armazenados na memória do controlador, que é um fator limitante do desenvolvimento do projeto.

4.4. Variáveis Analisadas

4.4.1. Produtividade das culturas

Foram avaliados durante o experimento as seguintes variáveis: massas de matéria fresca (MF) e seca (MS) das plantas obtidas por meio de balança de precisão de 0,01 g, número de folhas por planta (NF) , que foi determinada a partir da contagem do número de folhas totais presentes em cada planta em todas as unidades experimentais, comprimento da parte aérea (ALT), cada uma delas realizada a cada dez dias após o transplantio (DAT) com auxílio de uma régua milimétrica (cm).

A eficiência do uso da água pelas plantas foi calculada pela relação entre a produtividade da cultura (Y), por meio da massa fresca e pela lâmina de irrigação total (ITN) aplicada no ciclo (SILVA et al., 2012), conforme a equação 11 a seguir:

$$EUA = \frac{Y}{ITN} \tag{11}$$

Em que:

EUA – Eficiência do uso da água (kg m^{-3})

Y – Produtividade da cultura, (kg);

ITN – Lâmina de irrigação total aplicada por tratamento (m^3).

4.4.1. Custos de mão de obra

Foram analisados os períodos em que fosse necessária a presença humana, atuando no cultivo, quando comparado com o sistema autônomo e comparando-os para analisar os custos com cada um dos métodos.

4.4.2. Utilização de recursos

Foram analisados os gastos dos recursos hídricos e gastos elétricos de um sistema convencional com o sistema autônomo e comparados esses dados para estabelecer um gráfico comparativo dos métodos no uso de ambientes controlados.

4.5. Área foliar

Após a colheita e pesagem das amostras, foram retiradas suas folhas e verificada a

área foliar de cada amostra. Para isso foi utilizado o medidor de área foliar modelo LI 3100 da Licor, afim de determinar a área foliar (cm^2) em todos os tratamentos estudados.

4.6. Análise estatística

A análise estatística foi realizada primeiramente pela avaliação da normalidade dos dados e posteriormente pela análise de variância (ANOVA) a 5% de probabilidade. Observando a significância dos mesmos, estes foram submetidos ao teste de Tukey a 5% de probabilidade, através do programa SISVAR (FERREIRA, 2011).

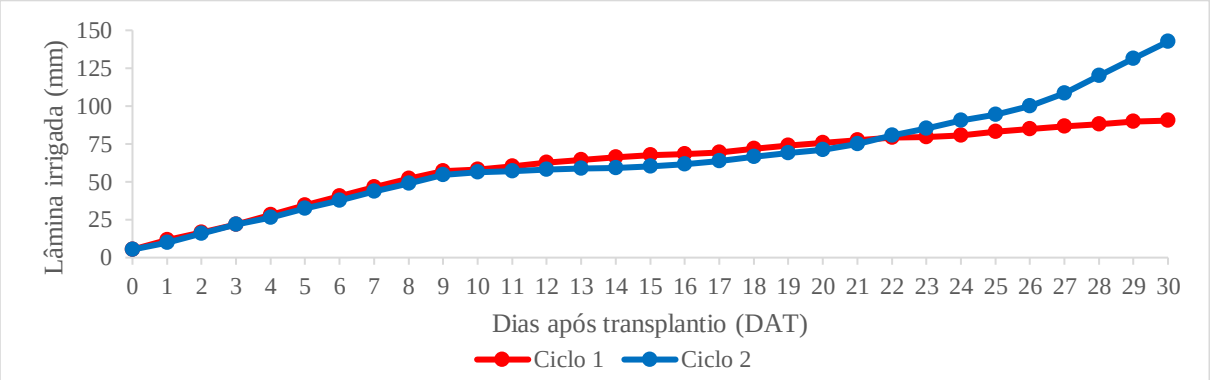
5. RESULTADOS E DISCUSSÃO

5.1. Manejos de irrigação automática

5.1.1. Manejo de irrigação automática via solo

As lâminas de irrigação aplicadas através dos dados obtidos pelo sensor ao longo dos dois ciclos da rúcula, podem ser observadas na Figura 10. O intervalo de umidade volumétrica lido pelo sensor ficou entre 76% e 82% [3], onde a cada decréscimo foram aplicados 0,8 L. Para o ciclo 1, foram aplicados 90,4 mm de água durante todo o ciclo, com lâmina média de 3,0 mm dia^{-1}. No ciclo 2 foram aplicados 142,5mm , com lâmina média de 4,7 mm dia^{-1}. Souza et al. (2019) em estudos sobre irrigação semiautomática na cultura do tomate, demonstraram a efetividade dos sensores de umidade do tipo TDR para o correto manejo da irrigação. Segundo Silva et al. (2020), os sensores de umidade do solo podem ser uma ferramenta importante para o correto manejo da irrigação, já que estes sensores podem irrigar no momento exato em que o solo está no processo de perda de umidade, repondo de maneira imediata o que foi consumido pela cultura.

Figura 10 - Lâmina aplicada no ciclo 1 e no ciclo 2 em milímetro.



Fonte: dados da pesquisa

Na Tabela 2 pode-se observar os valores de lâmina de irrigação aplicadas durante os ciclos de cultivo da cultura da rúcula. Os valores observados no primeiro experimento são menores que os observados por Cunha et al. (2013), onde estes autores obtiveram valores de evapotranspiração da cultura de 125 mm $ciclo^{-1}$ com uso da irrigação localizada, portanto a necessidade de uma menor aplicação de lâmina fica compreendida entre as diferentes condições

[3] Os valores lidos pelo sensor referem-se a porcentagem da capacidade de leitura do sensor e não necessariamente a umidade do solo em sí

de cultivo, já que os autores obtiveram tais valores em campo, o que demonstra a eficiência do IAS no uso da água. Para o segundo experimento, os valores foram superiores, devido a fatores construtivos do sensor, que gerou um erro na leitura realizada pelo controlador, gerando um excesso de irrigação em alguns períodos.

O erro da leitura no sistema IAS, foi ocasionado pelo próprio contato do sensor com o solo, pois este acabou sendo afetado pela oxidação, ação intrínseca dos metais, isso acarreta a necessidade de uma manutenção de limpeza do sensor, ou troca do material por um mais nobre, para evitar o acúmulo de resíduos de oxidação, que afeta a leitura do sensor, alterando seu real resultado, ou o que também pode ser feito, é adequar a programação a nova faixa de leituras, fazendo uma nova calibragem, que embora não seja o aconselhável por limitar a faixa de operação do controlador, mas em casos extremos, pode ser uma opção a ser considerada. Tal oxidação foi percebida no sensor utilizado, podendo assim, explicar a diferentes lâminas aplicadas no ciclo.

Tabela 2 - Dados de irrigação do sistema automático via solo

	Tempo irrigado (min)	Volume irrigado (L)	Lâmina irrigada (mm)
	---------- **Ciclo 1** ----------		
Total por tratamento	258,0	108,5	90,4
Média diária	8,6	3,62	3,0
	---------- **Ciclo 2** ----------		
Total por tratamento	534,00	171,0	142,5
Média diária	17,8	5,7	4,7

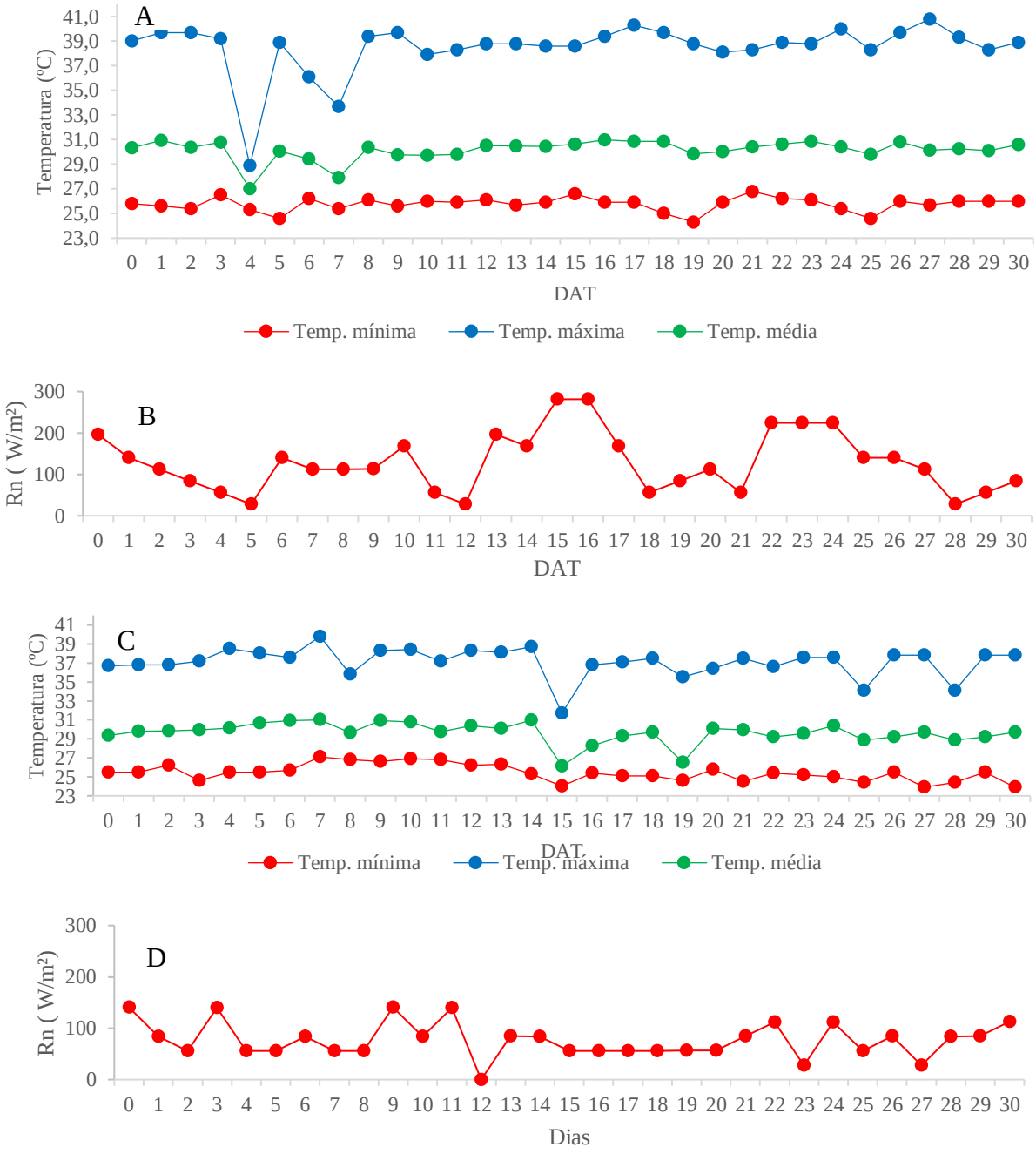
Fonte: dados da pesquisa

5.1.2. Manejo Automático da irrigação via clima

Para o IMC as temperaturas mínimas, médias e máximas obtidas no interior da casa de vegetação como também a radiação solar, podem ser observadas na Figura 11. As temperaturas máximas tiveram médias de 38,5°C, enquanto as médias foram de 30,2°C e as mínimas de 25,8°C para o ciclo 1. No ciclo 2, os valores para temperaturas máxima, média e mínima foram respectivamente 37,1ºC, 29,6ºC e 25,4ºC. A radiação solar obtida dentro do ambiente protegido teve valores médios de 128 W m^{-2} no ciclo 1 e 77 W m^{-2} no segundo ciclo. Com a obtenção destes dados foi possível determinar a evapotranspiração de referência (ETo), através do método de Hargreaves-Samani (ALLEN et al., 1998). Segundo Alencar, Sediyama e Mantovani (2015), o método de Penaman-Monteth FAO56 apresenta maior precisão para determinação da ETo, contudo a baixa disponibilidade de dados meteorológicos dentro do

ambiente protegido requer estimativas de ETo com a menor quantidade de dados possíveis, neste sentido, o uso da equação de Hargreaves-Samani, pode atender de maneira satisfatória esta demanda, apenas com dados de temperatura do ar, umidade relativa e radiação solar.

Figura 11 - Temperaturas máxima, média e mínima (A) e radiação solar (B) durante o ciclo 1 de produção e (C) e (D), para o ciclo 2 respectivamente.



Fonte: dados da pesquisa

Na Tabela 3 pode-se observar os valores de lâmina de irrigação aplicadas durante os ciclos de cultivo da cultura da rúcula e na Figura 12, pode ser observado a aplicação de lâmina durante ambos os ciclos. Para o ciclo 1, foram aplicados 95,9 mm de água, equivalentes

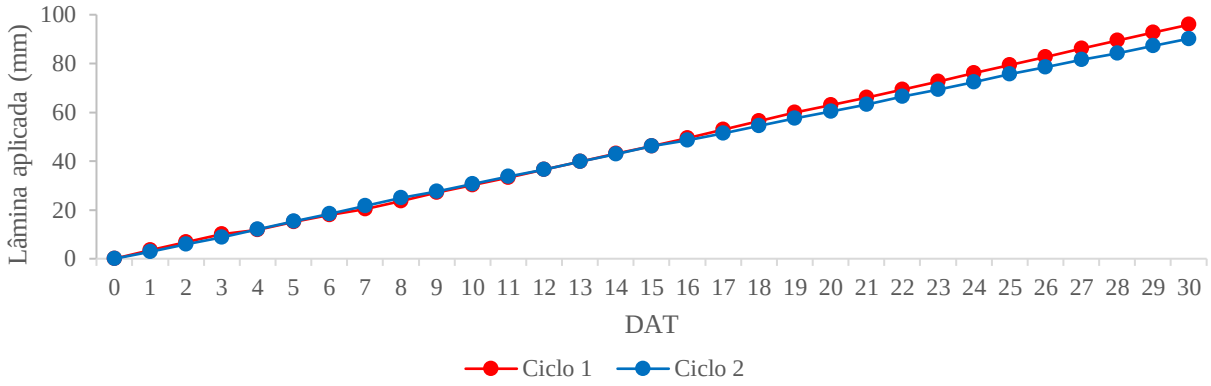
a 115,1 L durante todo o ciclo, com lâmina média de 3,2 mm dia^{-1}. No ciclo 2 foram aplicados 93,1 mm de água, equivalentes a 111,7 L durante todo o ciclo, com lâmina média de 3,1 mm dia^{-1}. Os valores observados no experimento estão acima dos valores observados por Cunha et al. (2013), neste sentido, possivelmente a região de cultivo podem ter influenciado nesta diferença, e provavelmente uma superestimativa da equação de Hargreaves-Samani na determinação da ETo.

Tabela 3 - Dados de irrigação do sistema automático via clima durante o experimento.

	Tempo irrigado (min)	Volume irrigado (L)	Lâmina irrigada (mm)
	---------- **Ciclo 1** ----------		
Total por tratamento	507,7	115,1	95,9
Média diária	16,9	3,8	3,2
	---------- **Ciclo 2** ----------		
Total por tratamento	477,4	108,2	90,2
Média diária	15,9	3,6	3,0

Fonte: dados da pesquisa

Figura 12 - Lâmina aplicada no sistema IAC nos ciclos



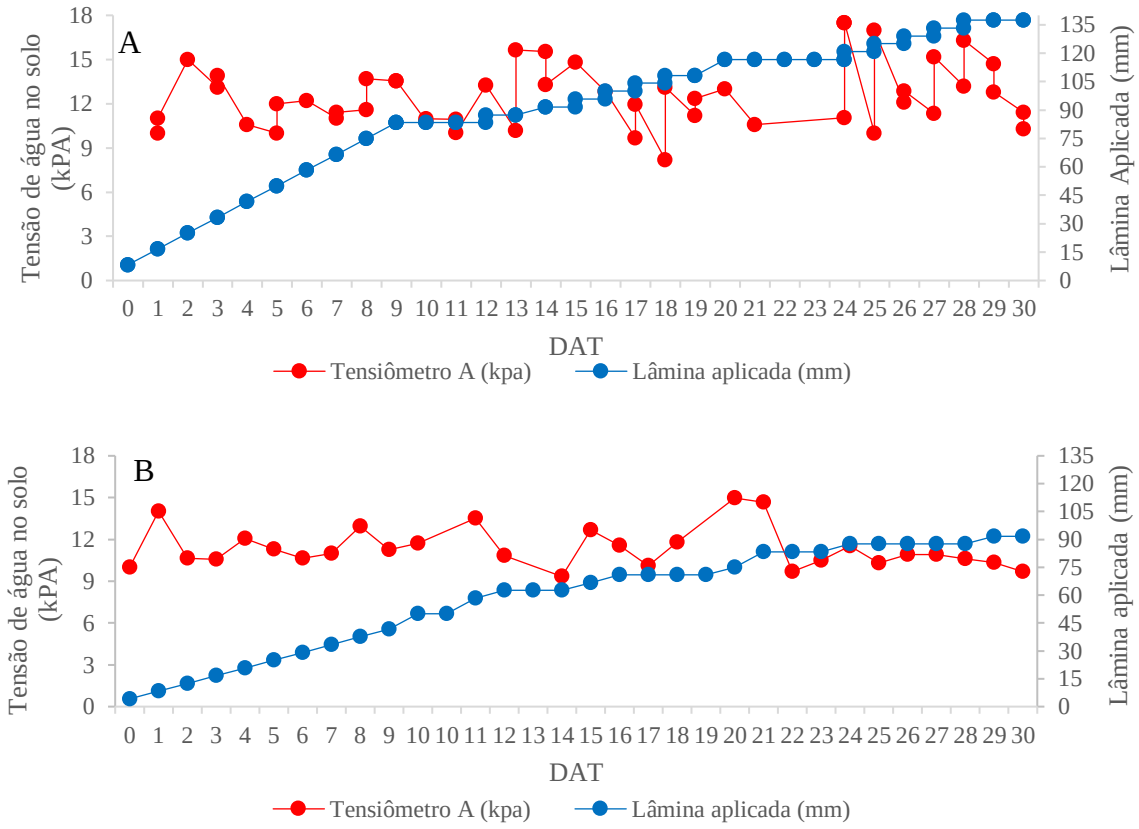
Fonte: dados da pesquisa

5.1.1. Manejo da irrigação manual via solo

O IMS realizado através do uso da umidade do solo, obtida via curva característica, pode ser observado na Figura 13. A frequência de irrigação foi realizada no período máximo de 2 dias, ou sempre que a tensão aproximava-se de 13 kPa, para evitar problemas de deficiência hídrica (MIRANDA et al., 2019; SILVA et al., 2015). Tal critério foi obtido de acordo com os resultados apresentados por Silva et al. (2015), onde estes autores observaram que tensões superiores a 15 Kpa reduzem a produtividade de hortaliças como a beterraba, sendo, portanto,

mantida uma tensão de água no solo inferior a estes valores para o presente estudo. Almeida et al. (2019) também observaram que tensões próximas a 18 kPa podem ser adequadas para delimitar o momento da irrigação.

Figura 13 - Potencial matricial e lâmina de irrigação acumulada no 1° ciclo (A) e 2° ciclo (B).



Fonte: dados da pesquisa

Os valores de lâmina irrigada acumulada na cultura foram de 137,5 mm no ciclo 1 e 93,8 mm no ciclo 2 (Tabela 4). A lâmina média diária para o IMS foi de 4,6 mm dia^{-1} para o ciclo 1 e 3,1 mm dia^{-1} para o segundo, no qual, os valores observados foram superiores aos observados por Cunha et al. (2018) com 106,1 mm ciclo^{-1}. As elevadas temperaturas observadas no período do presente estudo podem ter influenciado no aumento do consumo de água pelas plantas.

Tabela 4 – Dados médios de irrigação do sistema manual via solo durante o experimento.

	Volume irrigado (L)	Lâmina irrigada (mm)
	---------- **1° Ciclo** ----------	
Total por tratamento	165,0	137,5
Média diária	5,5	4,6

	---------- **2° Ciclo** ----------	
Total por tratamento	112,5	93,8
Média diária	3,8	3,1

Fonte: dados da pesquisa

5.1.1. Manejo de irrigação manual via clima

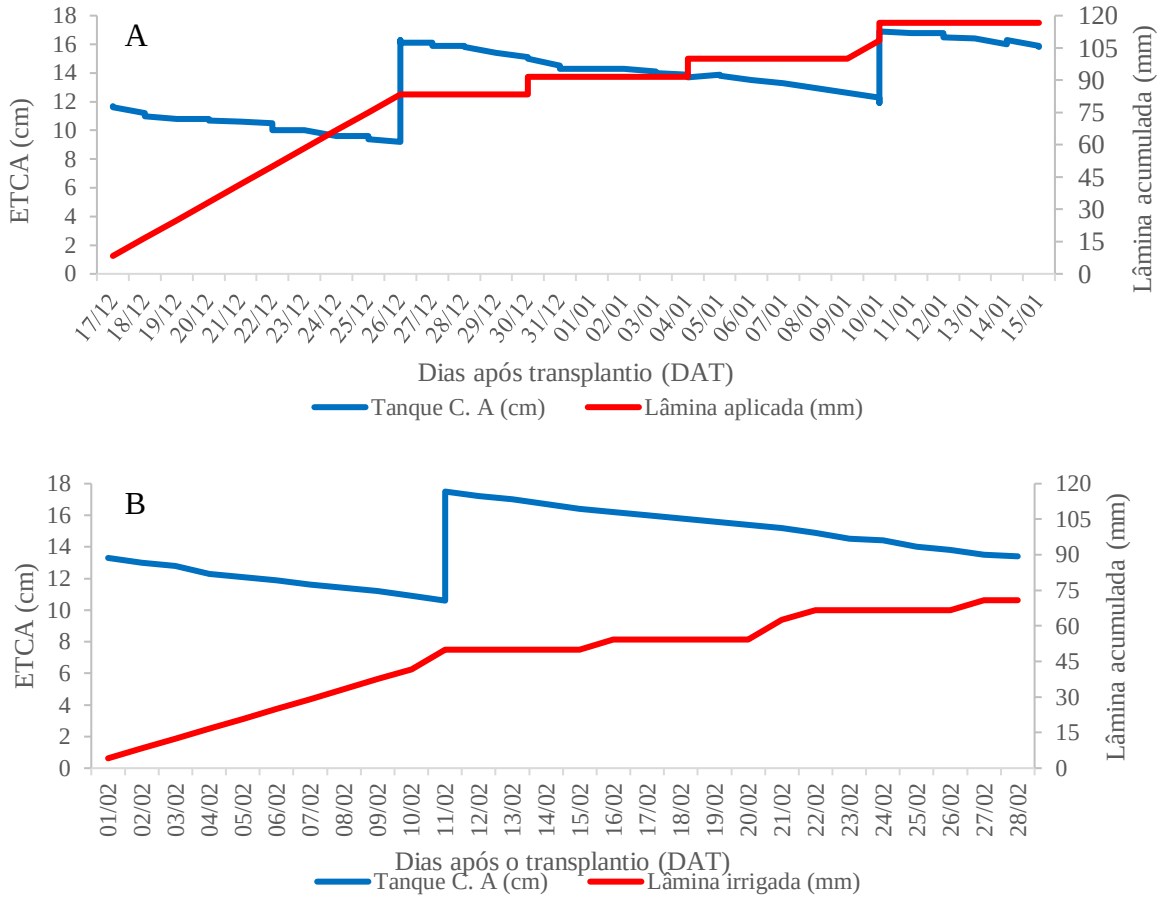
Para o manejo IMC, obteve-se uma lâmina de irrigação de 116,7 mm (Tabela 5), com consumo diário de 3,9 mm dia^{-1} no 1° ciclo (Figura 14A) na Figura, pode ser visto o nível do tanque classe A durante os ciclos e seu decaimento de acordo com a evaporação de cada dia, até o momento que foi necessário repor a água do tanque para aumentar o nível do tanque para continuar as leituras. No 2° ciclo os valores de ETc total foram de 75,0 mm $ciclo^{-1}$, com consumo diário de 2,5 mm dia^{-1} (Figura 14B). Os valores do 1° ciclo se aproximam dos obtidos no IMS e se assemelham aos valores obtidos por Cunha et al. (2018), possivelmente devido aos valores de ETc obtido serem semelhantes em relação a sua metodologia (ALLEN et al., 1998). No período de chuvas, em decorrência da redução de temperatura, onde o clima ficou mais ameno, foi possível observar uma variação nos dados do 2° ciclo , como o aumento da umidade e consequentemente, redução da evaporação do tanque classe A.

Tabela 5 - Dados de irrigação do sistema manual via clima durante o experimento.

	Volume irrigado (L)	Lâmina irrigada (mm)
	---------- **Ciclo 1** ----------	
Total por tratamento	140,0	116,7
Média diária	4,7	3,9
	---------- **Ciclo 2** ----------	
Total por tratamento	90,0	75,0
Média diária	3,0	2,5

Fonte: dados da pesquisa

Figura 14 - Nível do tanque classe A e lâmina aplicada no ciclo 1 (A) e no ciclo 2 (B)



Fonte: dados da pesquisa

5.2. Variáveis de crescimento

De acordo com a análise de variância pode-se observar que apesar de um alto coeficiente de variação, o fator manejo da irrigação apresentou efeito significativo para a variável altura de plantas aos 20 DAT ($p < 0,05$) e 30 DAT ($p<0,01$) no 1° ciclo de produção, como pode ser observado na Tabela 6. Possivelmente a disponibilidade constante de água ao longo do ciclo pode ter influenciado de maneira a diferenciar os manejos da irrigação a partir dos 20 DAT (MOLINE et al., 2015).

No 2° ciclo, todas as variáveis apresentaram efeito significativo ($p<0,01$) para o fator manejo da irrigação. Devido a diferenciação de cada método, possivelmente os valores de lâmina acumulada influenciaram neste fator. Tal fato foi observado por diversas pesquisas (CUNHA et al., 2018; MOLINE et al., 2015; CUNHA et al., 2013), o que comprova que a adição de água aumenta o vigor da cultura e altera o seu crescimento. Contudo, o estresse causado pelo excesso de água também pode ser um fator a ser considerado nesta avaliação,

conforme avaliado por Souza et al. (2019) em estudos sobre a profundidade do lençol freático e sua influência na produtividade da cultura da rúcula, onde os autores observaram que a água em excesso pode prejudicar o desenvolvimento desta cultura.

Tabela 6 - Análise e variância (quadrado médio) para a variável altura de plantas aos 10, 20 e 30 dias após o transplantio (DAT)

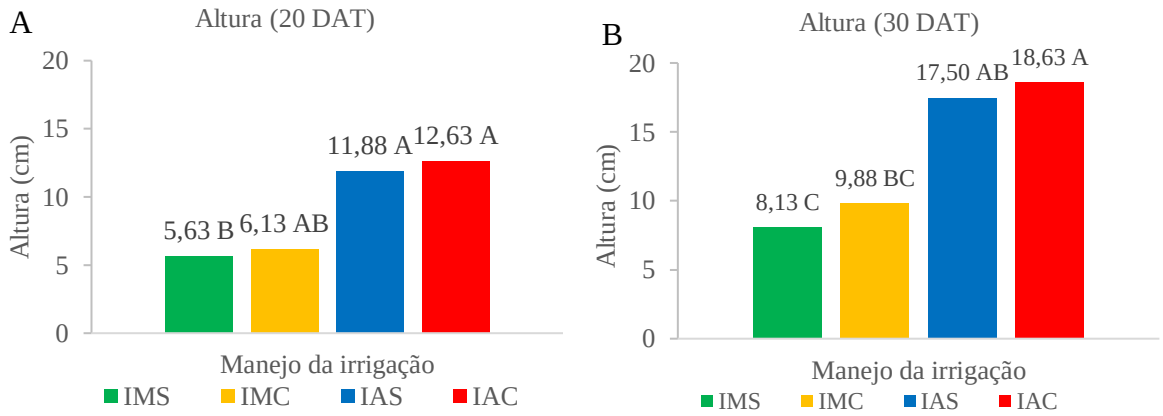
F.V.	---------- 1° Ciclo----------			
	G. L.	ALT 10 DAT	ALT 20 DAT	ALT 30 DAT
Bloco	7	$6,35^{NS}$	$26,28^{NS}$	$26,46^{NS}$
Tratamento	3	$3,75^{NS}$	83,53*	224,78**
Resíduo	21	3,98	17,84	36,11
C.V. (%)		38,97	43,74	44,41
F.V.	---------- 2° Ciclo ----------			
	G. L.	ALT 10 DAT	ALT 20 DAT	ALT 30 DAT
Bloco	7	0,80 NS	2,09 NS	4,42 NS
Tratamento	3	11,17**	84,70**	77,27 **
Resíduo	21	1,17	5,28	6,78
C.V. (%)		14,65	17,08	14,83

** e * - Significativo a 1 e 5% de probabilidade respectivamente; NS – não significativo

Fonte: dados da pesquisa

De acordo com os dados observados na Figura 14, para o 1° ciclo de produção, o IAC proporcionou maiores valores para a variável altura das plantas, com valores de 11,88 cm (Figura 15A) e 18,63 cm (Figura 15B) juntamente com o método IAS, que não diferem entre sí estatisticamente pelo teste de Tukey a 5% de probabilidade. O tratamento IMS apresentou os menores valores (5,63 cm) aos 20 DAT e (8,13 cm) e 30 DAT. Os valores observados no presente estudo, diferem dos apresentados por Moline et al. (2015) que obtiveram valores de até 25 cm para a altura da rúcula com 100% da reposição da ETc na região sul de Rondônia, Dias et al. (2019) também observaram valores acima dos encontrados nesta pesquisa, tais diferenças, possivelmente, possam estar relacionadas as diferentes condições de pesquisa e épocas de colheita de cada experimento.

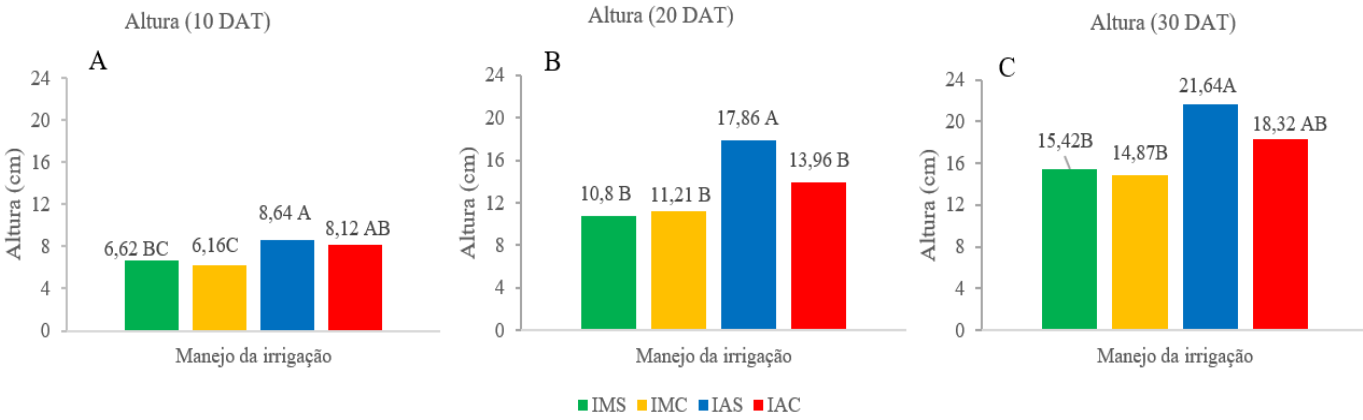
Figura 15 - Altura de plantas submetida ao teste de Tukey aos 20 DAT, DMS = 5,88 (A), 30 DAT, DMS = 8,37 (B) no ciclo 1.



*Letras iguais não diferem pelo teste de Tukey a 5%.

Os resultados do 2° ciclo para a variável altura de plantas podem ser observados na Figura 16. De acordo com os dados, pode-se observar que o IAS apresentou maiores valores para a variável estudada em todos os períodos de avaliação. Aos 20 DAT (Figura 16B) foi observado que o tratamento IAS obteve o maior valor (17,86 cm) dentre os tratamentos estudados, enquanto os tratamentos IAC, IMC e IMS não diferiram estatisticamente entre si.

Figura 16 - Altura de plantas submetida ao teste de Tukey aos 10 DAT, DMS = 1,50 (A), 20 DAT, DMS = 3,21 (B) e 30 DAT, DMS = 3,63 (C) no 2° ciclo da cultura da rúcula sob diferentes manejos de irrigação.



*Letras iguais não diferem pelo teste de Tukey a 5%.

Possivelmente a exigência por água neste período fenológico da cultura é menor (DOORENBOS e PRUITT, 1977), o que não acarretou diferenças entre estes tratamentos. Aos 30 DAT (Figura 16B), observou-se que para os manejos IAS e IAC foram obtidos os maiores valores (21,64 e 18,32 cm) para a variável estudada, não diferindo entre si ($p<0,05$). Os manejos

IMS, IMC e IAC não diferiram entre si, estatisticamente.

Apesar dos critérios de irrigação estabelecidos para a cultura da rúcula (MAROUELLI, MELO, BRAGA, 2017), o estresse hídrico pode ter ocasionado as diferenças nos tratamentos. Tal estresse pode ter sido causado pelo único horário em que as irrigações manuais eram realizadas, durante o dia (9 h da manhã), diferentemente das irrigações com sistema automático, que eram manejadas por pulsos, ao longo do dia, o que possivelmente pode estar relacionado aos maiores valores de produtividade observados no manejo automático. Diversos trabalhos tem debatido o uso da irrigação por pulsos, Maller et al. (2019) em estudos sobre a umidade do solo em irrigação por pulsos em um sistema por gotejamento, concluíram que a irrigação pulsante tende a distribuir a água no solo de maneira similar a irrigação contínua. Contudo, Almeida et al. (2015) em estudos com irrigação por pulsos, observaram redução do uso da água e aumento da produtividade da cultura da alface (*Lactuca sativa* L.), sendo esta uma cultura com elevado teor de água em sua constituição, semelhante a rúcula.

Para a variável número de folhas (NF) não foram observadas diferenças significativas entre os tratamentos estudados pelo teste F na probabilidade de 5% (Tabela 7) em ambos os ciclos de cultivo. Possivelmente o NF não diferiu entre os tratamentos estudados devido a manutenção da disponibilidade de água no solo em valores pré-estabelecidos pelos tratamentos. Trabalhos como os de Cunha et al. (2018) observaram aumento do NF de acordo com o incremento de água aplicado ao longo do ciclo, no qual a porcentagem de lâmina de água calculada foi maior ou menor que 100% da ETc. Possivelmente, a reposição da disponibilidade de água no solo em valores adequados é fator decisivo para a manutenção das folhas nas plantas, enquanto plantas em déficit hídrico adotam a redução de sua área foliar, através da diminuição do número de folhas, como estratégia para diminuir a superfície transpirante e o gasto metabólico (TAIZ et al., 2017; PINCELI e SILVA, 2012).

Tabela 7 - Análise e variância (quadrado médio) para a variável número de folha aos 10, 20 e 30 dias após o transplantio (DAT)

F.V.	---------- 1° Ciclo ----------			
	G. L.	NF 10 DAT	NF 20 DAT	NF 30 DAT
Bloco	7	0,71 ^{NS}	4,21 ^{NS}	20,24 ^{NS}
Tratamento	3	0,58 ^{NS}	11,58 ^{NS}	13,11 ^{NS}
Resíduo	21	1,84	4,55	8,95
C.V. (%)		29,37	28,47	44,41
F.V.	---------- 2° Ciclo ----------			
	G. L.	NF 10 DAT	NF 20 DAT	NF 30 DAT
Bloco	7	0,19 ^{NS}	0,78^{NS}	2,85 ^{NS}
Tratamento	3	0,12 ^{NS}	0,44^{NS}	7,08 ^{NS}
Resíduo	21	0,29	1,21	3,08
C.V. (%)		9,71	13,49	18,98

** e * - Significativo a 1 e 5% de probabilidade respectivamente; NS – não significativo

Fonte: dados da pesquisa

5.3. Variáveis de produção

De acordo com a análise de variância (Teste F), as variáveis área foliar (AF) e massa fresca da parte aérea (MFPA) apresentaram diferenças significativas ($p<0,05$) para os tratamentos estudados. Tal fato pode ter ocorrido devido a diferentes aplicações de água no solo (irrigação por pulsos e contínua) nos manejos da irrigação estudados. Trabalhos como os de Cunha et al. (2018), Moline et al. (2015) demonstraram diferenças significativas devido a variação das lâminas de irrigação para estas variáveis.

Tabela 8 – Análise e variância (quadrado médio) para a variável área folear (AF), massa fresca (MFPA) e massa seca (MSPA) da parte aérea aos 30 dias após o transplantio (DAT)

F.V.	---------- 1° Ciclo ----------			
	G. L.	AF	MFPA	MSPA
Bloco	7	13299,21 ^{NS}	108,35 ^{NS}	1,74 ^{NS}
Tratamento	3	48520,75 *	249,69 *	3,61 ^{NS}
Resíduo	21	11236,51	83,07	1,25
C.V. (%)		29,37	28,47	44,41
F.V.	---------- 2° Ciclo ----------			

	G. L.	AF	MF	MS
Bloco	7	6886,94 ^{NS}	21,89 ^{NS}	0,14 ^{NS}
Tratamento	3	33305,14**	203,31**	0,62 NS
Resíduo	21	4073,32	25,28	0,2
C.V. (%)		30,96	36,28	39,98

** e * - Significativo a 1 e 5% de probabilidade respectivamente, NS – não significativo

Fonte: dados da pesquisa

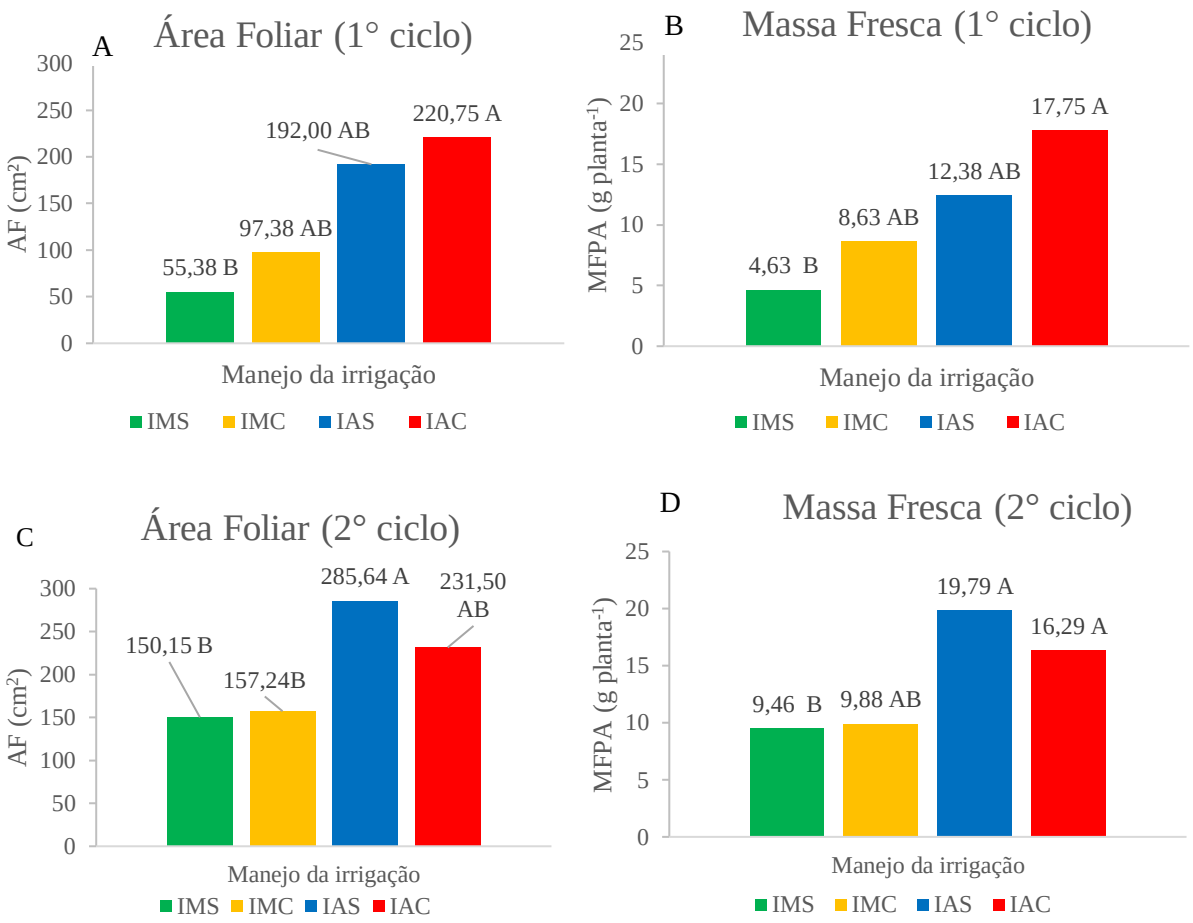
Na Figura 18 pode-se observar a cultura da rúcula após sua colheita no primeiro e segundo ciclo de produção. Os tratamentos com manejo da irrigação automático apresentaram uma maior massa em relação aos tratamentos com manejo da irrigação aplicados de maneira manual para o 1° ciclo . Tais resultados podem ser observados na Figura 17A, no qual o tratamento IMC foi o único que apresentou diferença estatística nos dados de AF, entre os tratamentos estudados ($p>0,05$), sendo este com o pior resultado.

No 2° ciclo, visualmente a área foliar entre os tratamentos automáticos é semelhante (Figura 18B), o que é confirmado pelos valores observados na Figura 17C, onde o IAC (231,50 cm²) e IAS (285,64 cm²) foram estatisticamente semelhantes ($p<0,05$) pelo teste de Tukey. No quesito viabilidade da produção, o manejo da irrigação através do uso de sistemas automáticos pode ser considerado satisfatório, pois apresentou valores superiores na maioria dos resultados estudados, comprovando a sua viabilidade no aspecto técnico de produção de hortaliças. Segundo Silva et al. (2020), a automação dos sistemas de irrigação é uma necessidade nos tempos atuais, devido a ocupação de produtores com outras atividades como agropecuárias, apicultura e etc. Contudo, quanto a questão econômica, estes autores são enfáticos em afirmar que a condição financeira de cada produtor, é um fator determinante para o uso de sistemas automáticos ou semiautomáticos (SOUZA et al. 2019). Outros insumos como o custo da água (FRIZZONE, 2007), fertilizantes (CAIXETA et al., 2017), e o próprio retorno econômico das hortaliças produzidas podem ser fatores decisivos para implementação da automação em sistemas irrigados.

Para a variável MFPA no 1° ciclo de produção (Figura 17B e 17D), observou-se que os tratamentos IAC, IAS e IMC foram estatisticamente semelhantes em ambos os ciclos avaliados. Diante disto, entende-se que o IMC pode ser uma opção viável para manejo da irrigação em situações de falta de recursos econômicos para implementação da automação dos sistemas de irrigação em pequenas propriedades, já que nas pequenas áreas o monitoramento da ETc pode ser realizadas com maior facilidade e praticidade através do uso do tanque Classe

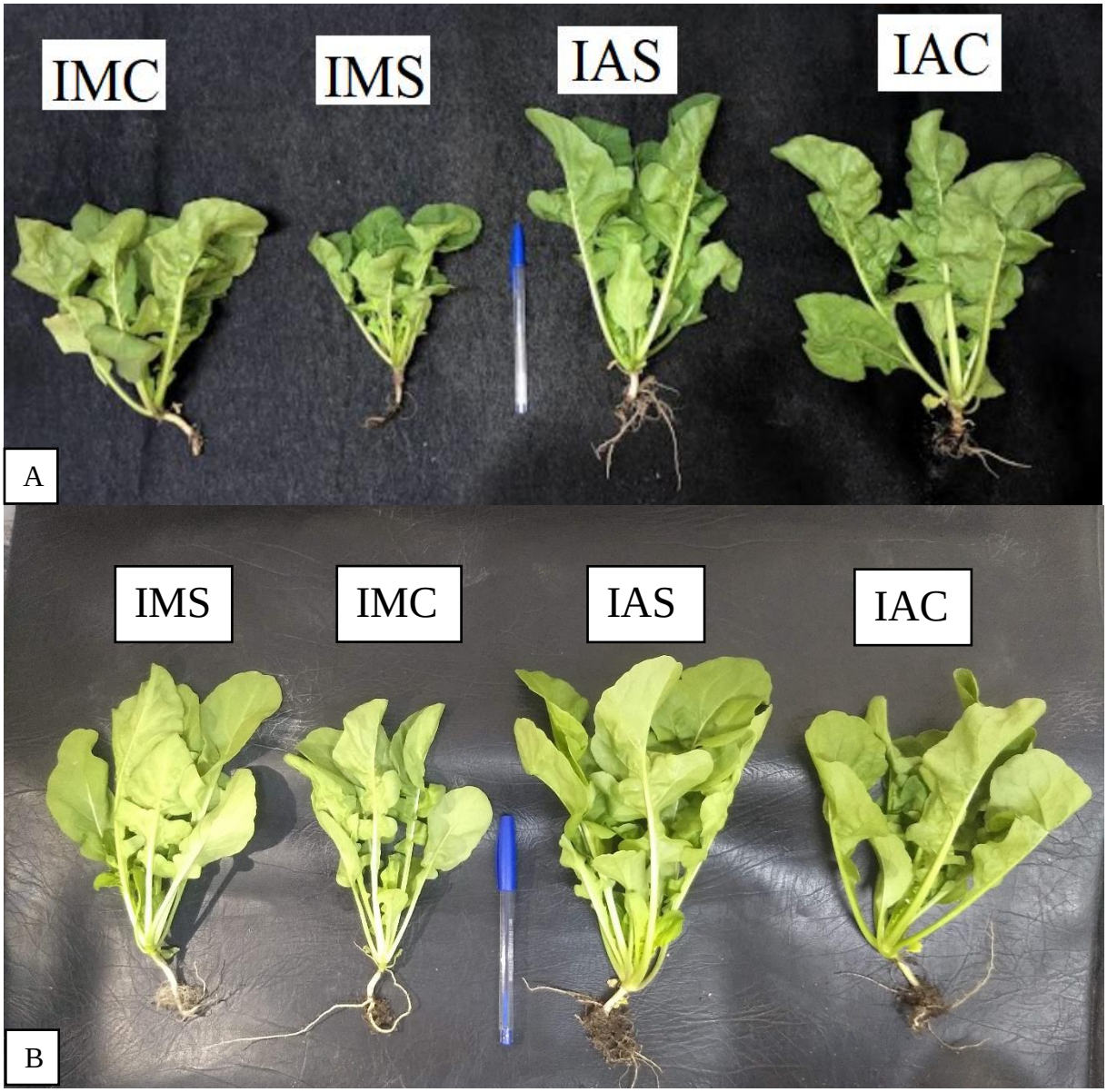
A. Moline et al. (2015) observaram valores superiores a 80 g em rúculas que receberam lâminas com 100% da ETc. Cunha et al. (2013) observaram valores de MF (17,03 g planta^{-1}) condizentes com os apresentados no presente experimento, o que demonstra a variabilidade da produção desta cultura aos métodos de produção utilizados.

Figura 17 - Variáveis de rendimento para área foliar no primeiro (DMS = 147,78) (A) e ciclo 2 (DMS = 88,98) (C), massa fresca da parte aérea no primeiro (DMS = 12,70) (B) e segundo (DMS=7,01) (D) ciclo para a cultura da rúcula sob diferentes manejos de irrigação.



Fonte: dados da pesquisa

Figura 18 - Massa fresca de rúcula submetida a diferentes manejos de irrigação para o 1° (A) e 2° ciclo (B).



Fonte: elaborado pelo autor

5.4. Eficiência do uso da água

Conforme pode-se observar na tabela 8, a eficiência do uso da água (EUA), foi influenciada ($p<0,05$) pelo fator manejo da irrigação apenas no 1° ciclo de produção. Conforme observado ao longo de todo o estudo, houve uma tendência a redução nas lâminas de irrigação aplicadas nos tratamentos estudados no 2° ciclo de produção, o que pode ter contribuído para aumento da EUA.

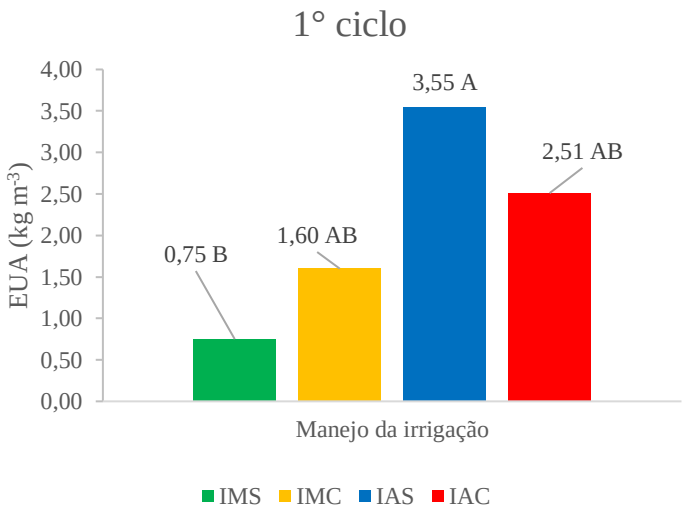
Tabela 8 - Análise e variância (quadrado médio) para a variável produtividade por quantidade de água utilizada.

F.V.	---------- EUA ----------		
	G. L.	Ciclo 1	Ciclo 2
Bloco	7	4,61 ^{NS}	0,96 ^{NS}
Tratamento	3	11,61*	2,1 ^{NS}
Resíduo	21	2,89	1,02
C.V. (%)		30,96	30,96

** e * - Significativo a 1 e 5% de probabilidade respectivamente, NS – não significativo

Para a variável EUA, observou-se que os tratamentos IAS (3,55 kg m^{-2} m^{-3}), IAC (2,51 kg m^{-2} m^{-3}) e IMC (1,60 kg m^{-2} m^{-3}) não diferiram entre si estatisticamente (Figura 19). Conforme Frizzone (2007), para desenvolver estratégias ótimas de irrigação é necessário utilizar relações entre a água aplicada e a produtividade, sendo estas denominadas funções de produção. Neste sentido, a EUA está relacionada ao modo de produção a qual a cultura está submetida, principalmente ao uso da água e seu método de aplicação, levando-se assim a considerar a automação como uma necessidade para aumentar a EUA (SILVA et al., 2020), o que pode ser observado para os tratamentos automáticos deste estudo, contudo o IMC deve ser levado em consideração em condições econômicas desfavoráveis para implementação da automação.

Figura 19 – Eficiência do uso da água para a cultura da rúcula (DMS = 2,37) no 1° ciclode produção sob diferentes manejos de irrigação



Fonte: dados da pesquisa

5.5. Análise econômica

Os quatro métodos apresentados tem semelhanças entre si e diferenças em relação a custos, por não necessitarem de materiais e mão de obra mais qualificada para montagem, os sistemas manuais tem custos mais baixos para instalação quando comparados aos automáticos, porém os automáticos dispensam a necessidade constante de mão de obra para cuidados da cultura. Pode ser observado na Tabela 9 os valores estipulados para os manejos automáticos e na Tabela 10 para os manuais, considerando os custos fixos iniciais e variáveis anuais dos manejos de irrigação.

Tabela 9 – Valores iniciais de custos relacionados aos manejos automáticos

	Automático via Clima		**Automático via Solo**	
	Material	**Total**	**Material**	**Total**
Custos fixos iniciais	controlador	R$50,00	controlador	R$50,00
	Sensores	R$20,00	Sensores	R$90,00
	válvula	R$45,00	válvula	R$45,00
	materiais elétricos gerais (infraestrutura)	R$100,00	materiais gerais (infraestrutura)	R$100,00
	mangueiras	R$70,00	mangueiras	R$70,00
	caixa d'agua	R$200,00	caixa d'agua	R$200,00
	sistema elétrico de alimentação	R$120,00	sistema de potência	R$120,00
	Instalação	R$2.200,00	Instalação	R$2.200,00
Custos variáveis de execução (anual)	Água	R$55,32	Água	R$55,32
	Energia	R$360,00	Energia	R$360,00
	Manutenção	R$2.200,00	Manutenção	R$2.200,00
	Total	**R$5.420,32**		**R$5.490,32**

Fonte: elaborado pelo autor

Tabela 10 – Valores iniciais de custos relacionados aos manejos manuais

	Manual via Clima		**Manual via Solo**	
	Material	**Total**	**Material**	**Total**
Custos fixos iniciais	caixa d'agua	R$200,00	caixa d'agua	R$200,00
	Tanque classe A	R$ 946,00	Tensiômetro	R$ 240,00
	Instalação (salário mínimo 2021)	R$550,00	Instalação (salário mínimo 2021)	R$550,00
Custos variáveis de execução (anual)	Água	R$55,32	Água	R$55,32
	Manutenção	R$1.100,00	Manutenção	R$1.100,00
	Total	**R$2.455,32**		**R$2.575,32**

Fonte: elaborado pelo autor

Nos manejos de irrigação automáticos não foram consideradas mão de obra mensal,

pois não há a necessidade de ter a presença humana frequentemente no local. Nos manejos manuais, faz-se necessário a presença de mão de obra constante.

Os custos de mão de obra de instalação utilizados para base de cálculo foram baseados no salário mínimo do ano do estudo desse trabalho. Para a instalação dos sistemas automáticos, pela sua complexidade foi considerado o dobro desse valor, devido a sua complexidade. Para os sistemas manuais foi considerado um salário mínimo. Para manutenção dos sistemas, foram considerados os mesmos valores, respectivamente.

Dentre os sistemas manuais pode-se observar que os custos com materiais, deve ser considerado, como por exemplo, a utilização de tanque classe A para a irrigação, este tem valor elevado comparado aos demais recursos necessários para os outros. Já o método via solo por tensiômetro tem valores mais atrativos. Entre os automáticos, não há uma discrepância entre valores para os métodos utilizados, porém esses valores irão divergir à medida que se aumenta a quantidade de sensores utilizados (SOUZA et al., 2019b) no solo, já que se faz necessário uma quantidade proporcional ao número de canteiros e casas de vegetação. Para o sistema via clima o aumento do número de sensores pode ser evitado pela própria estrutura do sistema que analisa temperatura e radiação no local, podendo ser utilizada de uma forma mais generalizada para mais de um local, caso estejam próximos e sejam semelhantes.

5.6. Comparação com sistemas existentes

Atualmente existem sistemas de irrigação automáticos sendo vendidos no mercado, porém que oferecem apenas o controle de irrigação por temporização. Isso significa que o usuário define previamente o tempo de irrigação e o sistema executa. O sistema fica responsável por irrigar por um determinado tempo e depois encerrar a irrigação, isso já é um avanço comparado ao sistema manual, porém quando comparado ao sistema apresentado, que identifica a necessidade da cultura, podemos ver os benefícios do sistema autônomo.

Os sistemas autônomos apresentados têm o diferencial de realizar leituras e execução de acordo com o caso de cada momento, por exemplo, caso haja chuvas e o sistema estiver implementado em campo aberto, é identificado a umidade do solo, evitando o acionamento da irrigação. No caso do manejo via clima, pelas leituras de temperatura, umidade do ar e radiação solar, o volume irrigado já será menor que os dias sem chuva. Nesse ponto, de economia de recursos, o sistema autônomo é mais vantajoso que o temporizado.

Em relação a comparação de custos, a diferença entre ele está na aquisição de sensores. Pois estes sensores de umidade têm custos de R$ 90,00 e o sensor de temperatura e

umidade com valores próximos a R$ 20,00 considerando-se estes, preços acessíveis. Quando comparado aos temporizadores do mercado, os valores desses irrigadores variam, porém os mais comuns custam: R$ 430,00 (modelo ESP-RZXe da Rain Bird), R$ 350,00 (modelo RPS46 da K Rain) e R$ 670,00 (modelo X-Core 601 da Hunter). Esses valores referem-se apenas aos controladores, lembrando que toda a infraestrutura se repete quando utilizados no sistema autônomo apresentado nesse projeto. Fica necessário a instalação de sistema hidráulico, circuitos elétricos, válvulas de acionamento, sistema de armazenagem de água e rede de comunicação. Os custos tornam-se semelhantes aos do projeto, porém este apresenta-se vantajosos por oferecer mais recursos que os já existentes.

6. CONCLUSÕES

Com base nos resultados obtidos, é possível concluir que os sistemas automáticos de irrigação podem ser uma ferramenta viável para aumentar a eficiência do uso da água, facilitando o manejo da água nos cultivos, reduzindo os custos com mão de obra e aumentando a produção.

A oxidação em sensores de umidade do solo é um problema real, tornando-se necessário manutenções para evitar problemas na leitura durante o ciclo produtivo, sendo necessário portanto, cautela ao utilizar apenas sensores de umidade como base para manejo da irrigação, devendo-se sempre quando possível, realizar o acompanhamento da evapotranspiração da cultura para possíveis correções.

Em relação ao manejo da irrigação de forma manual, o manejo da irrigação via clima com uso do tanque classe A, no presente estudo, apresentou o melhor desempenho, sendo portanto, uma opção quando não se pretende investir inicialmente em tecnologia e já se possui os equipamentos de obtenção de dados climáticos.

REFERÊNCIAS

AGUIAR, M. A., ESCOBEDO, J. F.; GALVANI, E. Influência da cultura do pimentão (Capsicum annuum L.) nos elementos ambientais em ambiente protegido. **Irriga**, Botucatu, v. 7, n. 3, p. 230, 2002.

ALENCAR, LEONIDAS P. DE; SEDIYAMA, GILBERTO C.; MANTOVANI, EVERARDO C. Estimativa da evapotranspiração de referência (ETo padrão FAO), para Minas Gerais, na ausência de alguns dados climáticos. **Engenharia Agrícola**, Jaboticabal, v. 35, n. 1, p. 39-50, fev. 2015.

ALLEN, R. G.; PEREIRA, L. S.; RAES, D.; SMITH, M. **Crop evapotranspiration: Guidelines for computing crop water requirements**. Rome: FAO, 1998. 300 p. (FAO – Irrigation and Drainage Paper N° 56).

ALMEIDA, A. C. S.; PUSCH, M.; BONIFÁCIO, J. S.; OLIVEIRA, F. C.; GEISENHOLFF, L.; BISCARO, G. A. Efeito da tensão crítica de irrigação e cobertura do solo sobre o cultivo de rabanete. **Agrarian**, Dourados, v. 12, n.45, p.308-317, 2019.

ALMEIDA, W. F.; LIMA, L. A.; PEREIRA, G. M. Drip pulses and soil mulching effect on American crisp head lettuce yield. **Engenharia Agrícola**, Jaboticabal, v. 35, n.6, p.1009-1018, 2015.

ARAGÃO, V. F.; FERNANDES, P.; GOMES FILHO, R. R., SANTOS NETO, A. M.; CARVALHO, C. D.; FEITOSA, H. O. Efeito de diferentes lâminas de irrigação e níveis de nitrogênio na fase vegetativa do pimentão em ambiente protegido. **Revista Brasileira de Agricultura Irrigada,** Fortaleza, v. 5, n. 4, p. 361-375, 2011.

BATISTA, A. V. A.; ALBIERO, D.; VIANNA, T. V. A.; MONTEIRO, L. A.; CHIODEROLI, C. A.; SOUSA, I. R. S.; AZEVEDO, B. M. Multifunctional Robot at low cost for small farms. **Ciência Rural**, Santa Maria, v. 47, p. 1-5, 2017.

BERNARDO, S.; MANTOVANI, E. C.; SILVA, D. D.; SOARES, A. A. **Manual de irrigação**. 9ª Ed. Viçosa, MG: Editora UFV, 2019, 545 p.

CAIXETA, M. M. A.; ALMEIDA, M. J.; WINDER, A. R. S.; DARIN, E. P.; BUSO, W. H. D. Desempenho da rúcula cultivada em diferentes modos de adubação. **Revista Mirante**, Anápolis - GO, v. 10, n. 2, p.191-200, 2017.

CALOU, V. B. C.; TEIXEIRA, A. S.; MOREIRA, L. C. J.; LIMA, C. S.; OLIVEIRA, J. B.; OLIVEIRA, M. R. R. . The use of UAVs in monitoring yellow sigatoka in banana. **Biosystems Engineering**, Amsterdã, v. 193, p. 115-125, 2020.

CARVALHO, D. F.; OLIVEIRA, L. F. C. **Planejamento e manejo da água na agricultura irrigada**. Viçosa: Ed. UFV, 2012. 240p.

CUNHA, F. F. et al. Irrigação de diferentes cultivares de rúcula no nordeste do Mato Grosso do Sul. **Water Resources and Irrigation Management**, Cruz das Almas, v. 2, n. 3, p. 131-141, 2013.

DIAS, M. S.; REIS, L. S.; SANTOS, R. S. H.; ALMEIDA, C. A. C.; PAES, R. A.; ALBUQUERQUE, A. W.; SILVA, F. A. Crescimento de plantas de rúcula em substratos e níveis de salinidade da água de irrigação. **Colloquium Agrarian**, Presidente Prudente, v. 15, n.4, , p. 22-30, 2019.

DOORENBOS, J.; PRUITT, W. O. **Crop water requirements**. Rome: FAO, 1977. 179 p. (FAO. Irrigation and Drainage Paper N° 24).

DOTA, M. A.; SANTOS, I. M.; CUGNASCA, C. E. Fusão de sensores na agricultura de precisão. In: CONGRESSO BRASILEIRO DE AGRICULTURA DE PRECISÃO – CONBAP 2010, 2010, Ribeirão Preto. **Anais...** Jaboticabal: SBEA, 2010, p.1-9.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA - EMBRAPA. **Sistema brasileiro de classificação de solos**. 3.ed. Brasília, 2013. 353p.

FELICIO, P. I. A.; RIBEIRO, R. S. F.; SILVA, A. O.; ARAUJO, J. C.; COSTA, R. N. T. Características físicas de cápsulas porosas para uso na irrigação localizada. **Irriga**, Botucatu, v. 4, p. 861-873, 2019.

FERREIRA, D. F. Sisvar: a computer statistical analysis system. **Ciência e Agrotecnologia**, Lavras, v. 35, n.6, p. 1039-1042, 2011.

FRIZZONE, J. A. Planejamento da irrigação com uso de técnicas de otimização. **Revista Brasileira de Agricultura Irrigada**, Fortaleza, v. 1, n. 1, p. 24-49, 2007.

HACKENHAAR, N. M.; HACKENHAAR, C.; ABREU, Y. V. Robótica na agricultura. **Interações**, Campo Grande, v. 16, n. 1, p. 119-129, 2015.

KLAR, A. E. **A água no sistema solo–planta–atmosfera**. 2.ed. São Paulo: Nobel, 1988. 408p

LIBARDI, P.L. **Dinâmica da água no solo**. 2ªed. São Paulo: Editora da Universidade de São Paulo, 2012, 341p

MALLER, A.; REZENDE, R.; FREITAS, P. S. L.; SERON, C. C.; HACHMANN, T.L. Umidade no perfil do solo com aplicações de água por pulsos utilizando gotejamento. **Revista Ciência Agronômica**, Fortaleza, v.50, n.2, pp.234-241, 2019.

MANTOVANI, E. C.; BERNARDO, S.; PALARTTI, L. F. **Irrigação:** princípios e métodos. Viçosa: Editora UFV, 2009. 328 p

MAROUELLI, W. A.; MELO, R. A. C.; BRAGA, M. B. **Irrigação no cultivo de brássicas**. Brasília: Embrapa Hortaliças, 2017. 25p. (Circular Técnica. 158).

MATARIC´, M. J. **Introdução à robótica**. São Paulo: editora UNESP, 2014. 367p.

MELO JÚNIOR, J. C. F.; GERVÁSIO, E. S.; ARMINDO, R. A. Sistema de automação para o manejo da subirrigação em ambiente protegido. **Irriga**, Botucatu, v. 18, n. 2, p. 337-350, 2013.

MIRANDA, J. R.; PEREIRA, G. M. Cultivo da beterraba sob diferentes tensões de água no solo. **Irriga**, Botucatu, v. 24, n.2, p.220-235, 2019.

MOLIN, J. P.; AMARAL. L. R.; COLAÇO, A. F. **Agricultura de precisão**. 1ed. São Paulo: Oficina de textos, 2015. 238p.

MONTELEONE, S.; MORAES, E. A.; FARIA, B. T.; AQUINO JÚNIOR, P. T.; MAIA, R. F.; TORRE NETO, A.; TOSCANO, A. Exploring the Adoption of Precision Agriculture for Irrigation in the Context of agriculture 4.0: The key role of internet of things. **Sensors**, Basel, v. 20, p. 1-32, 2020.

OLDONI, H.; BASSOI, L. H. Delineation of irrigation management zones in a Quartzipsamment of the Brazilian semiarid region. **Pesquisa Agropecuária Brasileira**, Brasília, v. 51, n.9, p.1283-1294, 2016.

OLDONI, H.; COSTA, B. R. S.; ROCHA JÚNIOR, R.C.; RABELLO, L. M.; BASSOI, L. H. Effects of size and sampling grid on the quality of apparent soil electrical conductivity maps. **Engenharia Agrícola**, Jaboticabal, v. 39, special issue, p.1-12, 2019.

PINCELI, R. P.; SILVA, M. A. Alterações morfológicas foliares em cultivares de cana-de-açúcar em resposta à deficiência hídrica. **Biosciense Journal**, Uberlândia, v. 28, n. 4, p. 546-556, 2012.

PURQUERIO, L.F. V.; TIVELLI, S. W. **Manejo do ambiente em cultivo protegido**. Campinas: Instituto Agronômico de Campinas, 2009, 11p. (Circular Técnica).

ROCHA NETO, O. C. ; TEIXEIRA, A. S. ; LEAO, R. A. O. ; MOREIRA, L.C.J. ; GALVÃO, L. S. Hyperspectral Remote Sensing for Detecting Soil Salinization Using ProSpecTIR-VS Aerial Imagery and Sensor Simulation. **Remote Sensing**, Basel, v. 9, p. 42, 2017.

ROCHA, M. G.; BARROS, F. M. M.; OLIVEIRA, S. R. M.; AMARAL, L. R. Biometric characteristics and canopy reflectance association for early-stage sugarcane biomass prediction. **Scientia Agrícola**, Piracicaba, v.76, n.4, p.274-280, 2019.

SANTOS, B. P.; SILVA, L. A. M.; CELES, C. S. F. S.; BORGES NETO, J. B.; PERES, B. S.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. A. F. Internet das coisas: da teoria à prática. *In:* SIQUEIRA, F. A.; LUNG, L. C.; GREVE, F. G. P.; FREITAS, A. E. S (org). **Minicursos/XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. 34 ed. Salvador: SBRC, 2016. p. 1-50. Disponível em: http://www.sbrc2016.ufba.br/downloads/anais/MinicursosSBRC2016.pdf. Acesso em: 20 mar. 2021.

SEGOVIA, J. F. O.; ANDRIOLO, J. L.; BURIOL, G. A.; SCHNEIDER, F. M. Comparação do crescimento e desenvolvimento da alface (*Lactuca sativa L.*) no interior e no exterior de uma estufa de polietileno em Santa Maria, RS. **Ciência Rural**, Santa Maria, v. 27, n. 1, p. 37-41, 1997.

SGANZERLA, E. **Nova Agricultura**: A fascinante arte de cultivar com os plásticos. 5.ed. Guaíba: Agropecuária, 1995. 342p.

SILVA, A. O.; SILVA, B. A.; SOUZA, C. F.; AZEVEDO, B. M.; BASSOI, L. H.; VASCONCELOS, D. V.; BONFIM, G. V.; JUAREZ, J. M.; SANTOS, A. F.; CARNEIRO, F. M. Irrigation in the age of agriculture 4.0: management, monitoring and precision. **Revista Ciência Agronômica**, Fortaleza, v. 51, p. 1-17, 2020.

SILVA, A. O.; SILVA, Ê. F.; BASSOI, L. H.; KLAR, A. E. Desenvolvimento de cultivares de beterraba sob diferentes tensões da água no solo. **Horticultura Brasileira**, Brasília, v. 33, n. 1, p. 12–18, 2015.

SILVA, A. O.; SOARES, T. M.; SILVA, E. F. F.; SANTOS, A. N.; KLAR, A. E. Consumo hídrico da rúcula em cultivo hidropônico NFT utilizando rejeitos de dessalinizadores em Ibimirim-PE. **Irriga**, Botucatu, v.17, p.114-125, 2012.

SOARES FILHO, R.; DA CUNHA, J. P. A. R. Agricultura de precisão: particularidades de sua adoção no sudoeste de Goiás–Brasil. **Engenharia Agrícola**, Jaboticabal, v. 35, n. 4, p. 689-698, 2015.

SOUZA, C. A.; ARAÚJO, B. A.; ROLIM, T. W. R.; TORRES, M. B.; SILVA, A. O.. Produção de rúcula em função da profundidade do lençol freático. **Revista Brasileira de Agricultura Irrigada**, Fortaleza, v. 13, n.5, p. 3656-3661, 2019a

SOUZA, C. F.; CONCHESQUI, M. ES; SILVA, M. B. Semiautomatic irrigation management in tomato. **Engenharia Agrícola**, Jaboticabal, v. 39, n. SPE, p. 118-125, 2019b.

SOUZA, C. F.; PIRES, R. C.; MIRANDA, D. B.; VARALLO, A. C. T. Calibração de sonda FDR e TDR para a estimativa da umidade em dois tipos de solo. **Irriga**, Botucatu, v. 18, n. 4, p. 597-606, 2013.

TAIZ, L.; ZEIGER, E.; MOLLER, I. MAX; MURPHY, A. **Fisiologia e Desenvolvimento Vegetal**. 6 ed. Porto Alegre: Artmed Editora, 2017. 858p.

TRANI, P.E.; RAIJ, B. van. Hortaliças. In: RAIJ, B. van; CANTARELLA, H.; QUAGGIO, J.A.; FURLANI, A.M.C. **Recomendações de Adubação e Calagem para o Estado de São**

Paulo, 2 ed. rev. ampl. Campinas: Instituto Agronômico & Fundação IAC, 1997. 285p. (Boletim Técnico, 100).

TSCHIEDEL, M.; FERREIRA, M. F. Introdução à agricultura de precisão: conceitos e vantagens. **Ciência Rural**, Santa Maria, v. 32, n. 1, p. 159-163, 2002.

VILLARES J. L. O.; SANTANA M. J.; FEITOSA NETO J. A.; MANCIN C. A.; RIBEIRO A. A. Evapotranspiração da cultura da rúcula cultivada no município de Uberaba, MG. **Horticultura Brasileira**, Brasília, v.29, n. 2, p.3469-3475, 2011.

ANEXO A – CODIGO FONTE COMENTADO DO SISTEMA DE IRRIGAÇÃO AUTÔNOMO VIA SOLO

```
/*SISTEMA DE CONTROLE DE IRRIGAÇÃO AUTÔNOMO
  //VERSÃO IA1 v0.4.1.3 //DATA 05/01/2021 // alterar em L350 a versão
  //VERSÃO DO ARDUINO: 1.8.13
  //MICROCONTROLADOR: NodeMCU 3.3V
  //INFORMAÇÕES ESPECÍFICAS SERÃO ENCONTRADAS NO FINAL DO CÓDIGO. DADOS COMO: Portas e funções, utilização de variáveis e funções, etc.

  /*ALTERAÇÕES
    1 - Adição de tempo de espera para infiltração
    2 -
  */

  // --- Definiçãod e bibliotecas ---
  #define BLYNK_PRINT Serial

  #include <ArduinoOTA.h> //biblioteca para utilizar OTA
  #include <DNSServer.h>  //Local DNS Server used for redirecting all requests to the configuration portal ( https://github.com/zhouhan0126/DNSServer---esp32 ) (WM)
  #include <EEPROM.h>     //controle de EEPROM
  #include <TimeLib.h>    //controle de tempo
  #include <WiFiUdp.h>    //comandos de UDP via WiFi
  #include <Wire.h>       //biblioteca responsável pela comunicação I2c

  //Bibliotecas exclusivas do ESP8266
  #include <BlynkSimpleEsp8266.h>   //biblioteca Blynk
  #include <ESP8266WebServer.h> //Local WebServer used to serve the configuration portal (WM)
  #include <ESP8266WiFi.h>      //controle do ESP8266
  #include <WiFiManager.h>          // WiFi Configuration Magic ( https://github.com/zhouhan0126/WIFIMANAGER-ESP32 ) >> https://github.com/tzapu/WiFiManager (ORIGINAL) (WM)

  // --- Mapeamento de Hardware---

  // --- FUNCIONAMENTO DO WIFI E BLYNK (VARIÁVEIS)---
  BlynkTimer timer;             //temporizador
  WidgetTerminal terminal(V8);  //Terminal virtual
  WiFiManager wifiManager;      //Wifi manager
  time_t prevDisplay = 0;       //controle de tempo

  //--- inicialização RTC online e UDP
  static const char ntpServerName[] = "a.st1.ntp.br";
  const int timeZone = -3;    // Brasil com horário de verão
  WiFiUDP Udp;
  unsigned int localPort = 8888;  // local port to listen for UDP packets
  time_t getNtpTime();

  //flag para indicar se foi salva uma nova configuração de rede
  bool shouldSaveConfig = false;

  // --- Blynk token
  char auth[] = "l3NsgQE-5bw1vRqWx_1NoN5-y_W9BgRs"; //projeto 1
  //char auth[] = "KLwIKI9ssZhWHfs9ySBHvQlcXTd3UpqH"; //projeto 2
  //char auth[] = "8rBrBLX1P-gHwWd832pwRw6A9T48Wp9W"; //testes

  // --- SSID e PASS
  //char ssid[] = "TW_Router";    //Regis
```

```
//char pass[] = "000111222"; //88839699
String ssid = WiFi.SSID(); //Pega a SSID gravada da ultima conexão
String pass = WiFi.psk();  //Pega a senha gravada da ultima conexão


// --- Mapeamento de Hardware ---
//DEFINIÇÃO DE BOTÕES
#define BOTAO_OK   D0  //botão OK na digital D0
#define SENSOR_  A0  //sensor na entrada A0 (como só tem uma entrada analógica)
#define VALV_    D1  //pino gpio da valvula
#define ERRO_    D2  //pino para alerta de erro na irrigação
#define RUN_     D3  // pino de controle de funcionamento

//DEFINIÇÃO DE MEMÓRIA EEPROM
#define VALORCALIB_0    130     // valor usado para local de memória EEPROM
#define VALORCALIB_100  131    // valor usado para local de memória EEPROM
#define UMIDADEAJUSTADA 132   // valor usado para local de memória EEPROM
#define TEMPO_IRRIGACAO 133   // valor usado para local de memória EEPROM
#define TEMPO_INFILTRA  134   // valor usado para local de memória EEPROM

//#define TEMPO_NOVA_IRRIGA 300000 // valor usado para definir intervalo de processo de atuação do controlador (em milissegundos)

// --- Constantes e Objetos ---

byte valorSensor = 0; //valores lidos dos sensores
byte ajusteSensor = 0; //valores ajustados dos sensores numa escala de 0 a 100%
byte umidadeRef = 0;   //umidade de referencia
byte Nirriga = 0;     //conta quantos ciclos de irrigação foram feitas no total, se + 4 e não finalizou, sinaliza um alerta
byte cont_pin = 0;    //conta qual a pino será acionado na vez
byte MaxCalib_0 = 0, MinCalib_100 = 0; //esse valor é definido para ajustar a calibração máxima e mínima, que serve de parametro para as funções
bool param_irrigaOK = false, param_infiltra = false, param_irrigando = 0, param_LerSensor = 0, param_teste = 0, param_run = 0, param_IrrigaManual = 0;

//PARAMETROS DE FUNÇÕES (respectivamente): iniciar calibração //alerta de erro // controle de irrigação // controle de leitura de sensores!
//long int tempo_atual = 0, tempo_nova_irriga = 0, tempo_irrigacao = 0;
byte verif_hora = 99, verif_minuto = 99, verif_segundo = 99; //verificadores para mudança de hora, minuto e segundo
byte cont_hora = 0, cont_minuto = 0, cont_segundo = 0, tempo_irriga = 0, cont_infiltra = 0, tempo_infiltra = 0;//contadores de tempo
int  cont_ti = 0;
String dataString = "";


// --- Parametros para Blynk ---
bool param_inicial = 0, param_calibrar = 0, param_erro = 0, param_reconexao = 0, param_reconBlynk = 0;


// LEGENDA dos TEMPOS
// tempo_atual (armazena o tempo atual para usar nos temporizadores) // tempo_nova_irriga (armazena o tempo para iniciar uma nova irrigação) // tempo_irrigacao (calcula o tempo de irrigação de cada canteiro)



// L88 --- FUNÇÕES DE INICIALIZAÇÃO ---
```

```
void  CalibraSensor();
int   tiraMediaLeitura();
void  leituraSensor();
void  config_WM();
void  chama_WM();
void  config_OTA();
void  sendNTPpacket(IPAddress &address);

//void digitalClockDisplay();

//
// L132 --- FUNÇÃO VOID myTimerEvent --- // A cada tempo, faz a atualização dos valores no aplicativo
void myTimerEvent() {
  // Pode-se enviar qualquer valor, a qualquer hora
  // OBS: não enviar mais de 10 valores por segundo.
  Blynk.virtualWrite(V0, umidadeRef);         //Envia para V0 o valor da umidadeRef gravada no sistema
  Blynk.virtualWrite(V1, ajusteSensor);     //Envia para V1 o valor da umidade lido em ajusteSensor
  //Blynk.virtualWrite(V2, acionam manual); //Envia para V2 o acionamento manual
  //Blynk.virtualWrite(V3, superv da bomba);//Envia para V3 o valor do funcionamento da valvula
  //Blynk.virtualWrite(V4, tempo);          //Envia para V4 o valor escolhido de tempo de irrigação
  //Blynk.virtualWrite(V5, param_calibrar); //Envia para V5 a variável de controle "param_calibrar" //v5 não será enviada aqui por motivos de segurança, só será enviada na função de calibração!
  //Blynk.virtualWrite(V6, param_erro);     //Envia para V6 o valor da umidade lido de param_erro //é enviado na função abaixo, pois o LED tem valores diferentes
  //Blynk.virtualWrite(V7, umidadeRef);     //Envia para V7 o valor da umidade lido de ajusteSensor
  //Blynk.virtualWrite(V8, terminal virtual);  //Envia para V8 dados para o terminal virtual
  Blynk.virtualWrite(V9, tempo_infiltra);   //Envia para V9 o valor de tempo_infiltra



  if (param_erro == 1) {
    Blynk.virtualWrite(V6, 255);  //Envia para V6 o valor da umidade lido de param_erro
  } else {
    Blynk.virtualWrite(V6, 0);  //Envia para V6 o valor da umidade lido de param_erro
  }

  if (digitalRead(VALV_) == LOW) {
    Blynk.virtualWrite(V3, 255);  //Envia para V3 o sinal de que a valvula está ligada
  } else {
    Blynk.virtualWrite(V3, 0);  //Envia para V3 o sinal de que a valvula está desligada
  }


}//FIM void myTimerEvent() {


// --- APLICATIVO V0 --- OBS: Alteração na umidade de referência
BLYNK_WRITE(V0) {
  int valorLido = param.asInt();
  if (valorLido > 100)
  {
    valorLido = 100;
  } else if (valorLido < 0)
  {
    valorLido = 0;
  }

  if (umidadeRef != valorLido)
```

```
                {
                 umidadeRef = valorLido;
                 EEPROM.write(UMIDADEAJUSTADA, umidadeRef);
                 if (EEPROM.commit())
                 {
                  dataString = "L136 UMIDADE LIDA PELO APLICATIVO E AJUSTADA: " +
String(umidadeRef);
                  Serial.println(dataString);
                  terminal.println(dataString);
                  terminal.flush();
                 } else
                 {
                  Serial.println("L133 ERROR! EEPROM não gravada!");
                  terminal.println("L133 ERROR! EEPROM não gravada!");
                  terminal.flush();
                 }
                }
               }//FIM BLYNK_WRITE(umidadeRef_V0){

               // L174 --- APLICATIVO V2 --- OBS: faz parametro de testes ser acionado e sistema funcionamento
para avaliação
               BLYNK_WRITE(V2) {
                int valorLido = param.asInt();
                if (valorLido == 1) //Essa função aciona manualmente a bomba de irrigação
                {
                 if (digitalRead(VALV_) == HIGH) //Essa função aciona manualmente a bomba de irrigação
                 {
                  digitalWrite(VALV_, LOW);  //desliga valvula de irrigação
                  Blynk.virtualWrite(V3, 255);  //Envia para V3 o sinal de que a valvula está ligada
                  Serial.println("Valvula acionada manualmente");
                  terminal.println("Valvula acionada manualmente");
                  terminal.flush();
                 }
                 else {
                  digitalWrite(VALV_, HIGH);  //desliga valvula de irrigação
                  Blynk.virtualWrite(V3, 0);  //Envia para V3 o sinal de que a valvula está ligada
                  Serial.println("Valvula desligada manualmente");
                  terminal.println("Valvula desligada manualmente");
                  terminal.flush();
                 }
                }
               }//FIM BLYNK_WRITE(V2){


               // --- APLICATIVO V4 --- OBS: Alteração do tempo de irrigação
               BLYNK_WRITE(V4) {
                int valorLido = param.asInt();
                if (valorLido > 255)
                {
                 valorLido = 255;
                } else if (valorLido < 0)
                {
                 valorLido = 0;
                }

                if (tempo_irriga != valorLido)
                {
                 tempo_irriga = valorLido;
                 EEPROM.write(TEMPO_IRRIGACAO, tempo_irriga);
                 if (EEPROM.commit())
```

```
                    {
                     dataString = "L136 TEMPO DE IRRIGAÇÃO LIDA PELO APLICATIVO E AJUSTADA: " +
String(tempo_irriga);
                     Serial.println(dataString);
                     terminal.println(dataString);
                     terminal.flush();
                    } else
                    {
                     Serial.println("L133 ERROR! EEPROM não gravada!");
                     terminal.println("L133 ERROR! EEPROM não gravada!");
                     terminal.flush();
                    }
                   }
                  }//FIM BLYNK_WRITE(umidadeRef_V4){

                  // --- APLICATIVO V5 --- OBS: Aciona calibração
                  BLYNK_WRITE(V5)
                  { int valorLido = param.asInt();
                   if (valorLido == 1)
                   {
                    param_calibrar = 1; //sabe que a calibração está ativada
                    //CalibraSensor();
                   }
                  }//FIM BLYNK_WRITE(V5){

                  //
                  // L160 --- APLICATIVO V6 --- OBS: Alerta se ocorreram muitas irrigações e não ouve mudança
no sistema
                  BLYNK_WRITE(V6)
                  { int valorLido = param.asInt();
                   if (valorLido == 0) //Essa função simplesmente reseta a variável erro, simbolizando que foi visto
pelo usuário!
                   {
                    param_erro = 0;
                   }
                  }//FIM BLYNK_WRITE(V6){

                  //
                  // L174 --- APLICATIVO V7 --- OBS: faz parametro de testes ser acionado e sistema funcionamento
para avaliação
                  BLYNK_WRITE(V7)
                  { int valorLido = param.asInt();
                   if (valorLido == 1) //Essa função simplesmente reseta a variável erro, simbolizando que foi visto
pelo usuário!
                   {
                    leituraSensor();  //faz leitura dos sensores, cada ciclo faz a leitura do sensor
                   }
                  }//FIM BLYNK_WRITE(V7){


                  // --- APLICATIVO V8 --- OBS: comandos vindo da serial
                  BLYNK_WRITE(V8) {
                   String  valorLido = param.asStr();
                   if (valorLido == "h") {
                    //--- cria uma string MENU que escreve um menu na tela do TV
                    char f_menu [] =  "0-Informações [info]\n"
                                      "1-Temperatura [temp]\n"
                                      "2-Umidade     [umid]\n"
                                      "3-Endereço IP [ ip ]\n"
                                      "4-Rede SSID   [ssid]\n"
```

```
                    "h-Ajuda       [ h ]\n"
                    "c-Limpar tela [clear]\n";
    Serial.println(f_menu);
    //terminal.clear();
    terminal.println(f_menu);
   }
   else if (valorLido == "ip") {
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
    terminal.print("IP: ");
    terminal.println(WiFi.localIP());
   }
   else if (valorLido == "ssid") {
    Serial.println("SSID: " + String(ssid));
    terminal.println("SSID: " + String(ssid));
   }
   else if (valorLido == "clear") {
    Serial.println("Limpando tela terminal virtual");
    terminal.clear();
   }
   else {
    Serial.print("Comando não reconhecido! Envie 'h' para ajuda" );
    terminal.print("Comando não reconhecido! Envie 'h' para ajuda");

   }
   // Assegurar que dados foram enviados
   terminal.flush();
  }//FIM BLYNK_WRITE(V8){


  // --- APLICATIVO V9 --- OBS: Alteração do tempo de infiltração
  BLYNK_WRITE(V9) {
   int valorLido = param.asInt();
   if (valorLido > 60)
   {
    valorLido = 60;
   } else if (valorLido < 0)
   {
    valorLido = 0;
   }

   if (tempo_infiltra != valorLido)
   {
    tempo_infiltra = valorLido;
    EEPROM.write(TEMPO_INFILTRA, tempo_infiltra);
    if (EEPROM.commit())
    {
     dataString = "L136 TEMPO DE INFILTRAÇÃO LIDA PELO APLICATIVO E AJUSTADA: "
+ String(tempo_infiltra);
     Serial.println(dataString);
     terminal.println(dataString);
     terminal.flush();
    } else
    {
     Serial.println("L133 ERROR! EEPROM não gravada!");
     terminal.println("L133 ERROR! EEPROM não gravada!");
     terminal.flush();
    }
   }
  }//FIM BLYNK_WRITE(V9){
```

```
// L330 --- void setup ---
void setup()
{
 // BEGIN necessários
 Serial.begin(9600);
 EEPROM.begin(512);
 delay(2000); // wait for console opening

 //=====================================
 // --- estabelece inicio do sistema
 Serial.println("\n\nL350 IA Primário v0.4.1.3 inicializado...");


 //=====================================
 // --- definindo portas e sinalizações de alerta
 pinMode(VALV_, OUTPUT); //definindo botão de entrada //no circuito está PULLDOWN
 digitalWrite(VALV_, HIGH);  //desliga valvula de irrigação
 //Serial.println("L190 Todas as valvulas e pinSensores configurados e iniciadas em DESLIGADO!");
 Serial.println("L190 Valvula iniciada em DESLIGADO!");

 pinMode(RUN_, OUTPUT); //definindo botão de entrada //no circuito está PULLDOWN
 digitalWrite(RUN_, HIGH);  //desliga valvula de irrigação

 pinMode(BOTAO_OK, INPUT); //botão para ok, reconfigurar WiFi (WM) //endereço de acesso 192.168.4.1
 Serial.println("L194 Botão OK e sensor definidos!");

 //  ////SERVEM APENAS PARA FORÇAR A CALIBRAÇÃO SO SENSOR (PARA TESTES)
 //     Serial.println("L197 Apagando Memoria EEPROM");
 //     EEPROM.write(130, 255); // VALORCALIB_0     max 255
 //     EEPROM.write(131, 0);   // VALORCALIB_100   min 0
 //     EEPROM.write(132, 50);   // UMIDADEAJUSTADA (0~100)
 //     if (EEPROM.commit())
 //         {
 //           Serial.println("L203 EEPROM gravada com sucesso");
 //         } else {
 //                 Serial.println("L205 ERROR! EEPROM não gravada");
 //               }


 //  Define para as variáveis os valores guardados na EEPROM
 MaxCalib_0   =  EEPROM.read(130);
 MinCalib_100 =  EEPROM.read(131);
 umidadeRef   =  EEPROM.read(132);   // inicia a variável umidade com o valor da EEPROM_132 ou seja, UMIDADEAJUSTADA
 tempo_irriga =  EEPROM.read(133);   // armazena o valor do tempo de irrigação
 tempo_infiltra = EEPROM.read(134);   // passa para a variável o valor do tempo de infiltração desejado

 Serial.println("L213 Lendo memória EEPROM");
 Serial.println(MaxCalib_0);
 Serial.println(MinCalib_100);
 Serial.println(umidadeRef);
 Serial.println(tempo_irriga);
 Serial.println(tempo_infiltra);
 Serial.println("L217 Memoria EEPROM valores vinculados com variáveis");
```

```
Blynk.virtualWrite(V0, umidadeRef); //Envia para V0 o valor da umidade gravada no sistema
//serve para iniciar o aplicativo com os valores do microcontrolador
Blynk.virtualWrite(V4, tempo_irriga);  //Envia para V4 o valor do tempo de irrigação
Blynk.virtualWrite(V5, param_calibrar);      //Envia para V5 o valor da variável controle
"param_calibrar"
Blynk.virtualWrite(V9, tempo_infiltra);      //Envia para V9 o valor da variável controle
"tempo_infiltra"



//=========================================================
// --- WIFI MANAGER
config_WM(); //configura o WiFi Manager //final do sketch


//=======================================
// --- CONEXÃO WIFI
Serial.print("L390 Conectando com: ");
Serial.println(ssid);

WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  //Serial.print(millis());
  if (millis() > 15000) { //espera 15 segundos tentando conectar, caso não consiga, continua, mesmo
offline
    Serial.println("L400 Não conectado ao Wifi...continuando o programa offline!");
    break;
  }
}

//=======================================
// --- CONEXÃO BLYNK
if (WiFi.status() == WL_CONNECTED) {
  Serial.print("\nL400 Dispositivo online no IP: ");
  Serial.println(WiFi.localIP());

  Blynk.config(auth);//inicializa servidor blynk

  //=======================================
  // --- CONEXÃO NTP UDP
  Serial.println("L410 Conectando a rtc NTP...");
  Udp.begin(localPort);
  setSyncProvider(getNtpTime);
  setSyncInterval(30 * 60); // tempo de sincronização em segundos (ideal a cada 10 ou 20 minutos)

  if (getNtpTime != 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso
seja zero, repete no próximo ciclo.
    param_reconexao = 1;
  } else {
    param_reconexao = 0;
  }
}// FIM if (WiFi.status() == WL_CONNECTED) {



//=======================================
// --- OTA ---
config_OTA(); //função pra configurar OTA //localizada no final do sketch
```

```
  //=====================================
  // --- ENVIO CONSTANTE DE INFORMAÇÕES PARA O BLYNK
  timer.setInterval(3500L, myTimerEvent); //atualização a cada 5000L = 5 segundos

}//FIM setup()


//Variáveis para Loop
//time_t prevDisplay = 0; //cria variável para controle de tempo
//WiFiManager wifiManager;//cria variável de controle no WiFi Mananger (WM)



// --- FUNÇÃO VOID LOOP ---
void loop()
{
  //---CICLOS INSTANTANEOS
  Blynk.run(); // Faz inicialização do Blynk
  timer.run(); // Initiates BlynkTimer
  chama_WM(BOTAO_OK);
  ArduinoOTA.handle();



  //ATUALIZAÇÃO DE REFERENCIA DE HORA
  if (now() != prevDisplay) { //atualiza valor mostrado se valor mudar //Atualiza prevDisplay e gera ciclos de 1 segundo e faz piscar LED de controle

    //CONTROLE DO TEMPO
    prevDisplay = now();      //prevDisplay recebe o valor atual do tempo now()
    verif_segundo = second();

    //--- CICLOS DE 1 HORA
    // --- verifica se passou um determinado tempo(hora ou minuto), para fazer uma nova leitura de temperatura e umidade

    //if (verif_hora != minute()) //para testes
    if (verif_hora != hour())
    { verif_hora = hour();

      Serial.println("L246 Entrando em um novo ciclo de leituras irrigação");
      terminal.println("L246 Entrando em um novo ciclo de leituras irrigação");
      terminal.flush();
      //ZERANDO VARIÁVEIS PARA COMEÇAR CICLO DE IRRIGAÇÃO=======================
      //tempo_nova_irriga = tempo_atual;
      param_irrigaOK = false;    //  variavel de controle de irrigação
      param_LerSensor = false;  //  variavel de controle de leitura de sensores
      Nirriga = 0;    //zera o número de contagens de irrigações (em outra parte se Nirriga > 5 gera um alerta de erro)
      //param_teste = 0;
    }// FIM if (verif_hora != hour())
    //--- FIM DE CICLOS DE 1 HORA


    //
    //---CICLOS DE 1 SEGUNDO
```

```
                //---Pisca LEDs --- RUN, WiFi e SD
                if (WiFi.status() != WL_CONNECTED) { //se conectado pisca LED RUN, se desconectado, pisca LED por tempo mais curto
                  digitalWrite(RUN_, 1); //liga LED RUN
                  delay(100);
                  digitalWrite(RUN_, 0); //desliga LED RUN
                  //Serial.println("L470 RUN ligado!");
                } else {
                  if (param_run == 1) {
                    digitalWrite(RUN_, 0); //led desligado
                    //Serial.println("L470 RUN desligado!");
                  } else {
                    digitalWrite(RUN_, 1); //led ligado
                    //Serial.println("L470 RUN ligado!");
                  }
                }

                if ((param_erro == 1) && (param_run == 1)) {
                  //if (param_erro == 1) {
                  digitalWrite(ERRO_, 1);//liga LED de erro
                  //Serial.println("L470 ERRO ligado!");
                } else {
                  digitalWrite(ERRO_, 0);//desliga LED de erro
                  //Serial.println("L470 ERRO desligado!");
                }

                if (param_run == 1) {
                  param_run = 0;
                } else {
                  param_run = 1;
                }


                //=======================================
                // --- se exite irrigação sendo feita, incrementa contador
                if (param_irrigando == 1) {
                  cont_ti++;
                  Serial.print("L459 cont_ti: ");
                  Serial.println(cont_ti);
                }

                //=======================================
                //--- Verifica se usuário solicitou uma calibração.
                if (param_calibrar == 1)//---responsável por verificar se usuário solicitou calibração
                {
                  CalibraSensor();
                  param_calibrar = 0;
                  Blynk.virtualWrite(V5, param_calibrar);  //Envia para V5 o valor da variavel param_calibrar e alertar que calibração finalizou
                }


                //=======================================
                //--- Verifica se dados foram enviados a Blynk ou desconectado.
                if (Blynk.connected() != 0) { //verifica se Blynk está conectado
                  if (verif_segundo % 30 == 0) { //a cada 30 segundos prepara dados a serem enviados
                    if (param_inicial == 0) {
                      terminal.println("L530 SISTEMA IA vIA1 v0.4.1.3 inicializado!"); terminal.flush(); //assegura se dados foram enviados com sucesso
                      param_inicial = 1; //muda o parametro para informar que dado foi enviado
```

```
                    param_reconBlynk = 1; //se inicializado não precisa enviar msg informando que conectou blynk
                  }
                  if (param_reconBlynk == 0) {
                    terminal.println("L530 [" + porzero_int(hour()) + "h" + porzero_int(minute()) + "min] ALERTA DE RECONEXÃO!"); terminal.flush(); //assegura se dados foram enviados com sucesso
                    param_reconBlynk = 1; //muda o parametro para informar que dado foi enviado
                  }
                  //      if (param_tv_leitura == 1) {
                  //        terminal.println(leituraString);
                  //        terminal.flush(); //assegura se dados foram enviados com sucesso
                  //        param_tv_leitura = 0; //muda o parametro para informar que dado foi enviado
                  //      }
                  //      if (param_tv_inicioirriga == 1) {
                  //        terminal.println(inicioirrigaString); //envia string de inicio de irrigação
                  //        terminal.flush(); //assegura se dados foram enviados com sucesso
                  //        param_tv_inicioirriga = 0; //muda o parametro para informar que dado foi enviado
                  //      }
                  //      if (param_tv_fimirriga == 1) {
                  //        terminal.println(fimirrigaString); //envia so os dados importantes
                  //        terminal.flush(); //assegura se dados foram enviados com sucesso
                  //        param_tv_fimirriga = 0; //muda o parametro para informar que dado foi enviado
                  //      }

                 }//FIM if (verif_segundo % 30 == 0)){
                }//FIM if ((Blynk.connected() != 0){
                else {
                 param_reconBlynk = 0;
                }


                //=====================================
                //--- Verificar situação se houve desconexão e tenta reconetar a Blynk e rtc NTP
                if (verif_segundo % 30 == 0) {
                 if ((WiFi.status() == WL_CONNECTED) && ((Blynk.connected() == 0) || (param_reconexao == 0))) //verifica a cada minuto se está conectado e blink conectado ou foi uma reconexão
                  {
                   //---se Blynk não conectado tenta conexão
                   if (Blynk.connected() == 0) {
                    Serial.println("L560 Conectando a BLYNK...");
                    Blynk.config(auth);//inicializa servidor blynk
                   }
                   //     Serial.print("L530 Valor de Blynk.connected(): ");
                   //     Serial.println(Blynk.connected());

                   //---recebe pacote de rtc NTP e atualiza hora e data
                   if (param_reconexao == 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso seja zero, repete no próximo ciclo.
                    Serial.println("L570 Conectando a rtc NTP...");
                    Udp.begin(localPort);
                    setSyncProvider(getNtpTime);
                    setSyncInterval(30 * 60); // tempo de sincronização em segundos (ideal a cada 10 ou 20 minutos)

                    if (getNtpTime != 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso seja zero, repete no próximo ciclo.
                     param_reconexao = 1;
                    } else {
                     param_reconexao = 0;
                    }
```

```
                    } //FIM if (param_reconexao == 0) {
                   }//FIM if ((WiFi.status() == WL_CONNECTED)
                  }//FIM if (second()%30 ==0){


                  // verifica a cada segundo se tempo de irrigação foi atingido
                  if (cont_ti > tempo_irriga) //verifica se o tempo de irrigação já passou //SE SIM = desliga valvula e passa pra próxima válvula
                  {
                   digitalWrite(VALV_, HIGH); //desliga a valvula solenóide de irrigação
                   dataString = "L289 [" + porzero_int(day()) + "/" + porzero_int(month()) + "/" + porzero_int(year()) + "] " + porzero_int(hour()) + 'h' + porzero_int(minute()) + "min";
                   dataString +=  "\nDESLIGANDO valvula. Ciclo de irrigação finalizado, aguardando infiltração";
                   Serial.println(dataString);
                   terminal.println(dataString);
                   terminal.flush();
                   Nirriga++; Serial.print("L301 Número de Nirriga="); Serial.println(Nirriga); //printando valor de Nirriga

                   cont_ti = 0;
                   param_irrigando = 0; //parametro de controle para saber se sistema está irrigando
                   param_LerSensor = false;  //fazendo parametro LerSensor falso, no proximo ciclo, faz-se leitura de sensores para ver se irriação foi o suficiente

                   param_infiltra = true;    //sinaliza que deve aguardar infiltrar;
                   Serial.println("Irrigação finalizada, aguardando infiltração");
                   terminal.println("Irrigação finalizada, aguardando infiltração");
                   terminal.flush();


                   if (Nirriga >= 5)
                   {
                    param_erro = 1;
                    Serial.println("L307 Alerta de ERRO! Muitas irrigações e pouco resultado!");
                    terminal.println("L307 Alerta de ERRO! Muitas irrigações e pouco resultado!");
                    terminal.flush();

                   }

                  } //FIM if (cont_ti > tempo_irriga)




                 } //FIM if (now() != prevDisplay) OU if (cont_segundo != second())
                 //FIM de CICLOS DE 1 SEGUNDO


                 //---CICLOS DE 1 MINUTO

                 //--- verifica se minuto mudou e atualiza
                 if (verif_minuto != minute()) {
                  verif_minuto = minute();

                  if (param_infiltra == true) { //esta infiltrando
                   cont_infiltra++;
```

```
                }
                if (cont_infiltra > tempo_infiltra) { //finalizou tempo de infiltrar
                  param_infiltra = false;
                  cont_infiltra = 0;
                }
                //=====================================
                //--- Verificar situação da conexão
                if (WiFi.status() != WL_CONNECTED) {
                  Serial.println("L540 Dispositivo desconectado, tentando conectar no WiFi...");
                  param_reconexao = 0; //parametro para forçar reconexão com UDP e Blynk
                  WiFi.begin(ssid, pass);
                }//FIM if (WiFi.status() != WL_CONNECTED) {


                //=====================================
                ////verifica se foi lido sensor e irrigado



                if ((param_irrigaOK == false) && (param_infiltra == false)) //se irrigação terminou e infiltração
terminou, da inicio a novo ciclo de leitura e irrigação
                {
                  if (param_LerSensor == false) //verifica se parametro ler sensor está satisfeito
                  { //SE SIM = pula este processo e vai para irrigação, SE NÃO = entra na função e faz leitura
                    //Serial.println("L461 Entrando em leituraSensor");
                    leituraSensor();  //faz leitura dos sensores, cada ciclo faz a leitura de um
                    Serial.println("L266 Leitura do sensor CONCLUIDA!");
                    param_LerSensor = true;
                  }//FIM if (param_LerSensor == false)


                  //=====================================
                  //verifica se parametro LerSensor foi satisfeito //SE DEVE IRRIGAR = entra na função
                  if (param_LerSensor == true) { //já foi feita a leitura dos sensores e preparando para irrigar
                    //IRRIGAÇÃO ======================================================
                    //Serial.println("L274 Iniciando irrigação");

                    if (ajusteSensor < umidadeRef)  //se umidade for menor que a desejada, entra na função
                    {
                      if (digitalRead(VALV_) != LOW)   //verifica se valvula ja foi acionada.
                      { //SE SIM = passa para proximo if,  SE NÃO = liga valvula e define parametros do timer
                        digitalWrite(VALV_, LOW); //liga a valvula solenóide de irrigação
                        dataString = "L281 [" + porzero_int(day()) + "/" + porzero_int(month()) + "/" +
porzero_int(year()) + "] " + porzero_int(hour()) + 'h' + porzero_int(minute()) + "min";
                        dataString +=  "\nACIONANDO  valvula: ";
                        Serial.println(dataString);
                        terminal.println(dataString);
                        terminal.flush();
                        param_irrigando = 1; //parametro de controle para saber se sistema está irrigando
                        cont_ti = 0; //contador do tempo de irrigação

                      }


                    }//FIM if(cont_pin >= N)
                  }//if (param_LerSensor == true)


                  if ((ajusteSensor >= umidadeRef) && (param_LerSensor == true))
                  { //verifica se todos os parametros foram satisfeiro, incluindo o parametro LERSensor, pois se
```

```
não, ele pode ficar preso na condição ideal e não fazer leitura de todos os sensores
                    dataString = "L530 [" + porzero_int(day()) + "/" + porzero_int(month()) + "/" + porzero_int(year()) + "] " + porzero_int(hour()) + 'h' + porzero_int(minute()) + "min";
                    dataString += "L535 IRRIGAÇÃO FINALIZADA! Níveis de umidade satisfeitos. Variáveis básicas resetadas!\n";
                    Serial.println(dataString);
                    param_irrigaOK = true; //quando processo de irrigação finaliza define param_irrigaOK positivo, assim define como irrigado
                    param_erro = 0;
                    Nirriga = 0; //as irrigações ocorreram bem.

                //      param_infiltra = true; //inicia tempo de espera para infiltrar
                //      Serial.println("Irrigação finalizada, aguardando infiltração");
                //      terminal.println("Irrigação finalizada, aguardando infiltração");
                //      terminal.flush();
                    } //FIM if ((ajusteSensor[0] >= umidadeRef))
                  } //FIM (param_irrigaOK == false))



                 } //FIM if (verif_minuto != minute()) {
                 //--- FIM de CICLOS DE 1 MIN




                }//FIM void loop


                // L332 --- FUNÇÃO VOID CalibraSensor ---
                void CalibraSensor()
                {
                 //DEFINIÇÃO E INICIALIZAÇÃO DE VARIÁVEIS
                 bool  confirma = 0;       //faz a verificação do botão OK
                 byte  AjusteAnalog = 0;   //armazena o valor ajustado das leituras (0~255)
                 int   mediaValSensor = 0; //armazena a média das leituras (0~1023)

                 //CALIBRAÇÃO PARA UMIDADE 0%==============================
                 //  digitalWrite(pinSensor[1], 1); //faz deslink do D1 com o sensor para leitura
                 //  digitalWrite(pinSensor[2], 1); //faz deslink do D2 com o sensor para leitura
                 //  digitalWrite(pinSensor[3], 1); //faz deslink do D3 com o sensor para leitura
                 //  digitalWrite(pinSensor[0], 0);  //faz link do D0 com o sensor para leitura // isso assegura que não haverá 2 relés ligados ao mesmo tempo
                 //  //a seguir no programa é necessário desfazer o link do relé para evitar problemas

                 //delay(300); //delay para esperar chaveamento do relé

                 //Serial.println("L348 Reles acionados");

                 Serial.println("L349 Insira o sensor para 0% e pressionar OK");
                 terminal.println("L349 Insira o sensor para 0% e pressionar OK");
                 terminal.flush();
                 delay(300); //delay para esperar chaveamento do relé

                 do {                          //DO WHILE //espera apertar o botão
                   confirma = digitalRead(BOTAO_OK);
                   delay(100);
                 } while (confirma == 0);           //DO WHILE
                 Serial.println("L355 Botão OK pressionado, aguardando liberar");
                 terminal.println("L355 Botão OK pressionado, aguardando liberar");
```

```
terminal.flush();

do {                          //DO WHILE //espera soltar o botão
  confirma = digitalRead(BOTAO_OK);
  delay(100);
} while (confirma == 1);          //DO WHILE
//Serial.println("L361 Botão OK liberado, entrando em for");

mediaValSensor = tiraMediaLeitura(SENSOR_); //calcula a média dos valores lidos do sensor

AjusteAnalog = map((mediaValSensor), 0, 1023, 0, 255); //lê o valor do sensor[i] distribui o valor lido entre 0 e 255 para melhor ajuste

if (EEPROM.read(VALORCALIB_0) != AjusteAnalog)
{ EEPROM.write(VALORCALIB_0, AjusteAnalog); //aplica esse valor na eeprom
  if (EEPROM.commit())
  { Serial.print("L409 EEPROM para VALORCALIB_0 gravada com sucesso: ");
    Serial.println(EEPROM.read(VALORCALIB_0));
    terminal.print("L409 EEPROM para VALORCALIB_0 gravada com sucesso: ");
    terminal.println(EEPROM.read(VALORCALIB_0));
    terminal.flush();
  } else {
    Serial.println("L411 ERROR! EEPROM para VALORCALIB_0 não gravada!");
    terminal.println("L411 ERROR! EEPROM para VALORCALIB_0 não gravada!");
    terminal.flush();
  }
} else {
  Serial.println("L417 EEPROM para VALORCALIB_0 já havia valor especificado!");
  terminal.println("L417 EEPROM para VALORCALIB_0 já havia valor especificado!");
  terminal.flush();
}


//CALIBRAÇÃO PARA UMIDADE 100%================================
confirma = 0;     //zerando variáveis para começar a leitura dos dados
mediaValSensor = 0;
Serial.println("L780 Insira o sensor para 100% e pressione OK!");
terminal.println("L780 Insira o sensor para 100% e pressione OK!");
terminal.flush();

do {                          //DO WHILE //espera apertar o botão
  confirma = digitalRead(BOTAO_OK);
  delay(100);
} while (confirma == 0);          //DO WHILE
Serial.println("790 Botão OK pressionado, aguardando liberar");
terminal.println("L790 Botão OK pressionado, aguardando liberar");
terminal.flush();

do {                          //DO WHILE //espera soltar o botão
  confirma = digitalRead(BOTAO_OK);
  delay(100);
} while (confirma == 1);          //DO WHILE
Serial.println("L395 Botão OK liberado");

mediaValSensor = tiraMediaLeitura(SENSOR_); //calcula a média dos valores lidos do sensor

//digitalWrite(pinSensor[0], 1); //faz deslink do D0 com o sensor para leitura // faz antes da gravação apenas para aproveitar o tempo de chaveamento

AjusteAnalog = map((mediaValSensor), 0, 1023, 0, 255); //lê o valor do sensor[i] distribui o valor
```

```
lido entre 0 e 255 para melhor ajuste
                dataString = "L402 Leitura do valor ajustado para gravar na EEPROM (0~255): " +
String(AjusteAnalog);
                Serial.println(dataString);
                terminal.println(dataString);
                terminal.flush();


                if (EEPROM.read(VALORCALIB_100) != AjusteAnalog)
                {
                 EEPROM.write(VALORCALIB_100, AjusteAnalog); //aplica esse valor na eeprom
                 if (EEPROM.commit())
                 { dataString = "L409 EEPROM para VALORCALIB_100 gravada com sucesso: " +
String(EEPROM.read(VALORCALIB_100));
                   Serial.println(dataString);
                   terminal.println(dataString);
                   terminal.flush();
                 } else {
                   Serial.println("L411 ERROR! EEPROM para VALORCALIB_100 não gravada!");
                   terminal.println("L411 ERROR! EEPROM para VALORCALIB_100 não gravada!");
                   terminal.flush();
                 }
                } else {
                 Serial.println("L417 EEPROM para VALORCALIB_100 já havia valor especificado!");
                 terminal.println("L417 EEPROM para VALORCALIB_100 já havia valor especificado!");
                 terminal.flush();
                }

                Serial.println("L419 Fim da Calibracao!");
                terminal.println("L419 Fim da Calibracao!");
                terminal.flush();
                //param_calibrar = 0; //mantem na variável que calibração finalizou

               }// FIM CalibraSensor()


               // L427--- FUNÇÃO INT tiraMediaLeitura() ---
               int tiraMediaLeitura (int TML_pin)
               {
                int TML_leitura = 0;  //armazena o valor o valor lido do sensor analógico
                int TML_soma = 0;     //armazena 5 valores de leitura para depois tirar uma média
                int TML_media = 0;

                //aqui faz 10 leituras do sensor para tirar uma média
                //Serial.println("L633 Entrando em FOR");
                for  (byte TML_cont = 0; TML_cont < 5; TML_cont++) //usa-se "c" para fazer a contagem desse
for, pois o "i" ja foi usado no for anterior
                {
                 TML_leitura = analogRead(TML_pin);  //faz leitura do valor e armazena
                 TML_soma = TML_soma + TML_leitura;    //soma com média
                 //Serial.print("L438 Valor da leitura do sensor analógico: ");
                 //Serial.println(TML_leitura);    //imprime valor lido
                 delay(200); //aguarda 200ms para proxima leitura
                }

                if (TML_soma < 5) //OBS: parece desnecessário esse procedimento, mas como havia uma divisão
onde era possível 0/5 ou 0/0 havia inscontancia nos dados
                {
                 TML_media = 0;
                }
```

```
  else {
    TML_media = TML_soma / 5; //retorna média da soma
  }
  Serial.print("L445 Valor da média lida  (0~1024): ");
  Serial.println(TML_media);
  return TML_media; //retorna 0

}//FIM int tiraMediaLeitura (int pin)


// --- FUNÇÃO VOID leituraSensor() ---
void leituraSensor()
{ //Serial.println("L462 Entrando em leituraSensor()");
  //LEITURA DOS SENSORES E IRRIGAÇÃO DE 1 A N
  int mediaValSensor = 0; //armazena a média das leituras
  mediaValSensor = tiraMediaLeitura(SENSOR_); //calcula a média dos valores lidos do sensor

  Serial.print("L821 Resultados da leitura de sensor");

  valorSensor = map((mediaValSensor), 0, 1023, 0, 255); //lê o valor do sensor distribui o valor lido entre 0 e 255 para melhor ajuste
  Serial.print("L481 Leitura ajustada      (0~255): ");
  Serial.println(valorSensor);
  if (valorSensor > MaxCalib_0) { //verificar se valores não ultrapassem de 100% que refere-se ao valor MaxCalib_0
    valorSensor = MaxCalib_0;
    Serial.println("L696 valorSensor ajustado para valor de MaxCalib");
  }

  if (valorSensor < MinCalib_100) { //verificar se valores não ultrapassem de 0% que refere-se ao valor MinCalib_100
    valorSensor = MinCalib_100;
    Serial.println("L701 valorSensor ajustado para valor de MinCalib");
  }

  //ajusta os valores do sensor para porcentagem de 0% a 100% de forma invertido pois quanto maior a leitura do sensor, mais seco o sistema está
  ajusteSensor = map(valorSensor, MinCalib_100, MaxCalib_0, 100, 0);  //map feito de forma invertida de 0-->255 para 100-->0 para se ajustar a leitura do sensor
  //Serial.println("L706 valorSensor ajustado para map");


  //IMPRIMINDO NA SERIAL
  Serial.print("L500 Umidade de Referência(0~100%): ");
  Serial.println(umidadeRef);
  Serial.print("L502 Umidade obtida       (0~100%): ");
  Serial.println(ajusteSensor);

}//FIM void leituraSensor()


// --- String porzero_int(int num_x) --- //*OBS: serve para transformar valor em numero de 2 digitos para jogar na data e hora
String porzero_int(int num_x)
{ String novo_num_x = "";
  if (num_x < 10)
    novo_num_x += '0';
  novo_num_x += String(num_x);
  return novo_num_x;
}//FIM String porzero_int(int num_x)
```

```
// --- String porzero_float(float num_x, num_y) --- //*OBS: serve para transformar valor em numero de 2 digitos para jogar na data e hora
String porzero_float(float num_x, int num_y) //num_x é a entrada //num_y é o número de casas que deve aparecer
{ String novo_num_x = "";
  if (num_x < 10)
    novo_num_x += '0';
  novo_num_x += String(num_x, num_y);
  return novo_num_x;
}//FIM String porzero_float(float num_x, int num_y)



// --- Códigos para funcionamento do RTC NTP
const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

time_t getNtpTime()
{
  IPAddress ntpServerIP; // NTP server's ip address

  while (Udp.parsePacket() > 0) ; // discard any previously received packets
  Serial.println("L1000 Enviando solicitação a NTP...");
  // get a random server from the pool
  WiFi.hostByName(ntpServerName, ntpServerIP);
  Serial.print(ntpServerName);
  Serial.print(": ");
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("L1000 Resposta NTP recebida");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }

  Serial.println("L1020 SEM RESPOSTA DE NTP :-(");
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
```

```
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
//FIM time_t getNtpTime()


// --- WIFI MANAGER

// --- CONFIGURAÇÃO WIFI MANAGER
void config_WM() {
  // *obs: só precisa dessas funções pois não vai acionar o WM no início, apenas quando for apertado o botão
  //declaração do objeto wifiManager
  //WiFiManager wifiManager;

  //utilizando esse comando, as configurações são apagadas da memória
  //caso tiver salvo alguma rede para conectar automaticamente, ela é apagada.
  //wifiManager.resetSettings();

  //callback para quando entra em modo de configuração AP
  wifiManager.setAPCallback(configModeCallback);
  //callback para quando se conecta em uma rede, ou seja, quando passa a trabalhar em modo estação
  wifiManager.setSaveConfigCallback(saveConfigCallback);

}

// --- void chama_WM(int WM_pin) //Códigos para funcionamento do WiFi Manager (WM)
//=========================================================
void chama_WM(int WM_pin) {
  //gera a cada loop uma nova variável para armazenar os valores
  //WiFiManager wifiManager;  //gerava um envio demasiado de msg pra serial, pois a cada vez criado era enviada uma msg

  //se o botão foi pressionado
  if ( digitalRead(WM_pin) == 1 ) {
    //WiFiManager wifiManager;  //gerava um envio demasiado de msg pra serial, pois a cada vez criado era enviada uma msg
    unsigned long WM_tempodelay = millis();
    while ((digitalRead(WM_pin) == 1) && ((millis() - WM_tempodelay) < 5000) ) {
      Serial.print("Pino pressionado!");
      Serial.println(millis() - WM_tempodelay);
      delay(100);
    }

    if ((millis() - WM_tempodelay) > 5000) {
      Serial.println("Iniciando WiFi Manager!"); //tenta abrir o portal
      terminal.println("Iniciando WiFi Manager!"); //envia so os dados importantes para terminal virtual blynk
      terminal.flush(); //assegura se dados foram enviados com sucesso
```

```
                WiFi.disconnect();
                delay(1000);

                wifiManager.startConfigPortal("WiFi_Manager", "WEB123456789"); //entra no portal toda vez, ao inves de tentar se conectar a uma rede anterior.
                wifiManager.setMinimumSignalQuality(10); // % minima para ele mostrar no SCAN
                wifiManager.setRemoveDuplicateAPs(true); //remover redes duplicadas (SSID iguais)
                wifiManager.setConfigPortalTimeout(20); //timeout para o ESP nao ficar esperando para ser configurado para sempre
                wifiManager.setConnectTimeout(30);

              }
             }
            }


            //callback que indica que o ESP entrou no modo AP
            void configModeCallback (WiFiManager * myWiFiManager)
            {
             Serial.println("Entrou no modo de configuração");
             Serial.println(WiFi.softAPIP()); //imprime o IP do AP
             Serial.println(myWiFiManager->getConfigPortalSSID()); //imprime o SSID criado da rede
            }

            //callback que indica que salvamos uma nova rede para se conectar (modo estação)
            void saveConfigCallback () {
             Serial.println("Configuração salva");
             Serial.println(WiFi.softAPIP()); //imprime o IP do AP
            }


            // --- OTA ---
            //---CONFIGURAÇÃO OTA
            void config_OTA() {
             // A porta fica default como 3232
             // ArduinoOTA.setPort(3232);

             // Define o hostname (opcional)
             ArduinoOTA.setHostname("TW_ESP8266_IA1");

             // Define a senha (opcional)
             ArduinoOTA.setPassword("WEB123456789");

             // É possível definir uma criptografia hash md5 para a senha usando a função "setPasswordHash"
             // Exemplo de MD5 para senha "admin" = 21232f297a57a5a743894a0e4a801fc3
             // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

             //define o que será executado quando o ArduinoOTA iniciar
             ArduinoOTA.onStart( startOTA ); //startOTA é uma função criada para simplificar o código

             //define o que será executado quando o ArduinoOTA terminar
             ArduinoOTA.onEnd( endOTA ); //endOTA é uma função criada para simplificar o código

             //define o que será executado quando o ArduinoOTA estiver gravando
             ArduinoOTA.onProgress( progressOTA ); //progressOTA é uma função criada para simplificar o código

             //define o que será executado quando o ArduinoOTA encontrar um erro
             ArduinoOTA.onError( errorOTA );//errorOTA é uma função criada para simplificar o código
```

```
  //inicializa ArduinoOTA
  ArduinoOTA.begin();
}


//funções de exibição dos estágios de upload (start, progress, end e error) do ArduinoOTA
//=======================================================================
//---MENSAGENS OTA
void startOTA()
{
  String type;

  //caso a atualização esteja sendo gravada na memória flash externa, então informa "flash"
  if (ArduinoOTA.getCommand() == U_FLASH)
    type = "flash";
  else  //caso a atualização seja feita pela memória interna (file system), então informa "filesystem"
    type = "filesystem"; // U_SPIFFS

  //exibe mensagem junto ao tipo de gravação
  Serial.println("Iniciando atualização " + type);
}
//=======================================================================
//---exibe mensagem fim
void endOTA()
{
  Serial.println("\nFIM");
}
//=======================================================================
//---exibe progresso em porcentagem
void progressOTA(unsigned int progress, unsigned int total)
{
  Serial.printf("Progresso: %u%%\r", (progress / (total / 100)));
}
//=======================================================================
//---caso aconteça algum erro, exibe especificamente o tipo do erro
void errorOTA(ota_error_t error)
{
  Serial.printf("Erro[%u]: ", error);

  if (error == OTA_AUTH_ERROR)
    Serial.println("Falha no inicio");
  else if (error == OTA_BEGIN_ERROR)
    Serial.println("Begin Failed");
  else if (error == OTA_CONNECT_ERROR)
    Serial.println("Falha na conexão");
  else if (error == OTA_RECEIVE_ERROR)
    Serial.println("Falha de recebimento");
  else if (error == OTA_END_ERROR)
    Serial.println("Falha na finalização");
}
```

ANEXO B – CODIGO FONTE COMENTADO DO SISTEMA DE IRRIGAÇÃO AUTÔNOMO VIA CLIMA

```
/*SISTEMA  DE CONTROLE DE IRRIGAÇÃO AUTÔNOMO

  //VERSÃO IA2 v0.4.4.3 //DATA 027/12/2020 // alterar em L350 a versão

  //VERSÃO DO ARDUINO: 1.8.13

  //MICROCONTROLADOR: NodeMCU 3.3V

  //INFORMAÇÕES ESPECÍFICAS SERÃO ENCONTRADAS NO FINAL DO CÓDIGO. DADOS COMO: Portas e funções, utilização de variáveis e funções, etc.


  /*ALTERAÇÕES

    1 - ADICIONADO +2 TEMPOS PARA IRRIGAÇÃO

    2 - ALTERAÇÕES NO APLICATIVO

*/


// --- Definiçãod e bibliotecas ---

#define BLYNK_PRINT Serial


#include <ArduinoOTA.h> //biblioteca para utilizar OTA

#include <DHT.h>        //biblioteca responsável pela comunicação com o sensor DHT22

#include <DNSServer.h>  //Local DNS Server used for redirecting all requests to the configuration portal ( https://github.com/zhouhan0126/DNSServer---esp32 ) (WM)

#include <EEPROM.h>     //controle de EEPROM

#include <SD.h>         //biblioteca responsável pela comunicação com o Cartão SD

#include <TimeLib.h>    //controle de tempo

#include <WiFiUdp.h>    //comandos de UDP via WiFi

#include <Wire.h>       //biblioteca responsável pela comunicação I2c
```

```
//Bibliotecas exclusivas do ESP8266

#include <BlynkSimpleEsp8266.h>   //biblioteca Blynk

#include <ESP8266WebServer.h> //Local WebServer used to serve the configuration portal (WM)

#include <ESP8266WiFi.h>      //controle do ESP8266

#include <WiFiManager.h>                // WiFi Configuration Magic ( https://github.com/zhouhan0126/WIFIMANAGER-ESP32 ) >> https://github.com/tzapu/WiFiManager (ORIGINAL) (WM)

// --- Mapeamento de Hardware---


#define DHTPIN    D4    // pino de dados do DHT será ligado no D2 do esp

#define DHTTYPE   DHT11 // tipo do sensor

#define CS_PIN    D8    // pino ligado ao CS do módulo SD Card

#define WM_RESET  D0    // pino de reset do sistema de WiFi (WM)

#define VALV      D1    // pino de controle da valvula solenóide

#define ERROSD    D2    // pino de controle de erro do micro SD //(D10 = Tx)

#define RUN       D3    // pino de controle de funcionamento    //(D9 = Rx)



//DEFINIÇÃO DE MEMÓRIA EEPROM

#define HORADEIRRIGAR       100 //hora 1 de irrigar   // valor usado para local de memória EEPROM 1 byte

#define MINUTODEIRRIGAR     101 //minuto 1 de irrigar // valor usado para local de memória EEPROM 1 byte

#define HORADEIRRIGAR2      102 //hora 2 de irrigar   // valor usado para local de memória EEPROM 1 byte

#define MINUTODEIRRIGAR2      103 //minuto 2 de irrigar // valor usado para local de memória EEPROM 1 byte

#define HORADEIRRIGAR3      104 //hora 2 de irrigar   // valor usado para local de memória EEPROM 1 byte
```

```
#define MINUTODEIRRIGAR3      105 //minuto 3 de irrigar // valor usado para local de memória EEPROM 1 byte

#define NIRRIGA               106 // valor usado pra definir o número de irrigações // valor usado para local de memória EEPROM 1 byte


#define IRRADIACAO            110 //+111 +112 +113  // valor usado para local de memória EEPROM 4 bytes

#define FATORDELTA            115 //+116 +117 +118  // valor usado para local de memória EEPROM 4 bytes

// --- Variáveis e Objetos ---

//--- DHT ---

DHT dht(DHTPIN, DHTTYPE);

float dht_umid = 0;   //variável que guarda o valor de umidade do DHT no ciclo // é float por os dados chegam em código binário

float dht_temp = 0;   //variável que guarda o valor de umidade do DHT no ciclo // é float por os dados chegam em código binário


//--- ARMAZENAMENTO DE DADOS ---

float   temp_max = 0.0f, temp_min = 99.0f, temp_media = 0.0f, temp_delta = 7.5;//variáveis de armazenamento de valores de temperatura máximo e mínimo de cada ciclo

float   bd_temp[24] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; // vetor de armazenamento de dados de temperatura

float   umid_max = 0.0f, umid_min = 99.0f, umid_media = 0.0f; //variáveis de armazenamento de valores de temperatura máximo e mínimo de cada ciclo

float   bd_umid[24] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; // vetor de armazenamento de dados de temperatura

String  dataString = "", leituraString = "", inicioirrigaString = "", fimirrigaString = ""; //variável que guarda o que será escrito no SD

bool    shouldSaveConfig = false; //flag para indicar se foi salva uma nova configuração de rede //edfinido por WiFiManager (WM)


// --- TEMPORIZADORES ---
```

```
unsigned int   cont_ti_auto = 0, cont_ti_manual = 0;   // tempo calculado para realizar irrigação da cultura e seu contador medido em segundos;

unsigned int   tempo_ti_auto = 0, tempo_ti_parcial = 0; //receberá valores em segundos

byte   tempo_ti_manual = 0; //auto guarda valores de segundos e manual guarda dados de unidades de minutos

byte  verif_hora = 99, verif_minuto = 99, verif_segundo = 99; //verifica hora, minuto e segundos para os processos do sistema


// --- PARAMETRIZADORES E CONTADORES ---

//---controle de processo

byte cont_bd = 0;  //contador usado nos processos de verificar mudança no tempo //inicializa de 99 para não confundir 0 na primeira leitura

bool param_irrigar_manual = 0, param_irrigar_auto = 0,  param_modo_irriga = 0, param_run = 0, param_reconexao = 0; //p_irrigar = se deve irrigar ou não //p_run = liga pisca LED // p_reconexao = serve para reconectar ao rtc e blynk após uma desconexão


//---parametros do aplicativo BLYNK

byte   param_hora1 = 0, param_minuto1 = 0, param_hora2 = 0, param_minuto2 = 0, param_hora3 = 0, param_minuto3 = 0;

byte  param_Nirriga = 0;

bool   param_erro = 0, param_inicial = 0, param_reconBlynk = 0, param_tv_leitura = 0, param_tv_inicioirriga = 0, param_tv_fimirriga = 0; //parametro de erro na gravação SD e de envio para Terminal virtual blynk

float param_irradiacao = 10.3f, param_delta = 0.05f, param_temp = 0.0f, param_umid = 0.0f, param_verifica = 0.0f; //coloca valores iniciais para não zerar algumas das funções


int startAt = 0, stopAt = 0; //valores iniciais para o Time Blynk do irrigador

char tz[] = "America/Fortaleza"; //TZone do Timer Blynk


//---WIFI E BLYNK---

BlynkTimer timer;

WidgetTerminal terminal(V7);
```

```
WiFiManager wifiManager;

//char auth[] = "l3NsgQE-5bw1vRqWx_1NoN5-y_W9BgRs"; //projeto 1
char auth[] = "KLwIKI9ssZhWHfs9ySBHvQlcXTd3UpqH"; //projeto 2
//char auth[] = "8rBrBLX1P-gHwWd832pwRw6A9T48Wp9W"; //testes

//char ssid[] = "TW_Router";    //Regis
//char pass[] = "000111222";    //88839699
//char ssid[] = "Regis Oi fibra 2.4g";
//char pass[] = "88839699";
String ssid = WiFi.SSID();
String pass = WiFi.psk();

//--- inicialização RTC online e UDP
static const char ntpServerName[] = "a.st1.ntp.br";
const int timeZone = -3;    // Brasil com horário de verão

WiFiUDP Udp;
unsigned int localPort = 8888;  // local port to listen for UDP packets
time_t getNtpTime();

// --- FUNÇÕES DE INICIALIZAÇÃO ---
void  gravarSD();
void  config_WM();
void  chama_WM();
void  config_OTA();
void  LerSensor_DHT11();
```

```
String porzero_int();

String porzero_float();

String escreve_datastring();

String escreve_datastring_diario();

void sendNTPpacket(IPAddress &address);




//void digitalClockDisplay();

// L132 --- FUNÇÃO VOID myTimerEvent --- // A cada tempo, faz a atualização dos valores no aplicativo

void myTimerEvent() {

  // Pode-se enviar qualquer valor, a qualquer hora

  // OBS: não enviar mais de 10 valores por segundo.



  //Blynk.virtualWrite(VX, Tinicial, Tfinal, TimeZone, Dias da semana); //serve apenas para dar idéia do modelo de envio de dados // todos são dados int, porém Dias da semana é um vetor do tipo char

  Blynk.virtualWrite(V0, ((param_hora1 * 60 + param_minuto1) * 60), stopAt, tz); //Tempo(segundos) = Hora*60*60 + Minuto*60 + Segundos

  //Blynk.virtualWrite(V1, param_erro);        //Envia para V1 -> erro de SD //não usa essa linha, pois o valor precisa ser 0~255 e param_erro é 0~1;

  Blynk.virtualWrite(V2, param_irradiacao);   //Envia para V2 -> valor de irradiação

  Blynk.virtualWrite(V3, param_delta);          //Envia para V3 -> valor do parametro DELTA, junção dos fatores (Kc*KL*A)/(Ea*n*Qg) onde: Kc(coef da cultura)/KL(coef localizada)/A(área da planta)/Ea(rendimento da irrigação/n(numero de gotejadores por planta)/Qg(vazão por gotejador)

  Blynk.virtualWrite(V4, param_temp);         //Envia para V4 -> temperatura de hora em hora

  Blynk.virtualWrite(V5, param_umid);         //Envia para V5 -> umidade de hora em hora

  Blynk.virtualWrite(V6, tempo_ti_auto / 60); //Envia para V6 -> tempo calculado para irrigação //enviado durante o calculo de ti

  //Blynk.virtualWrite(V7, TerminalVirtual);  //Envia para V7 -> comandos do terminal virtual
```

```
  //Blynk.virtualWrite(V8, param_modo_irriga);//Envia para V8 -> valor de irrigação manual, se ligado ou desligado

  Blynk.virtualWrite(V9, param_irrigar_manual); //Envia para V9 -> valor de ti da irrigação manual

  Blynk.virtualWrite(V10, tempo_ti_manual);    //Envia para V10 -> valor de ti da irrigação manual

  //Blynk.virtualWrite(V11, param_led/grafico de acionamento);          //Envia para V11 -> sinalização do led serve tbm para saber no grafico quando foi acionado

  Blynk.virtualWrite(V12, param_Nirriga);    //Envia para V12 -> valor do número de irrigações

  Blynk.virtualWrite(V13, tempo_ti_parcial / 60);    //Envia para V13 -> valor de ti parcial da irrigação. *OBS(tempo em segundos e divide por 60 para ter valor em minutos)

  Blynk.virtualWrite(V14, ((param_hora2 * 60 + param_minuto2) * 60), stopAt, tz); //Tempo(segundos) = Hora*60*60 + Minuto*60 + Segundos (segundo horário de irrigação)

  Blynk.virtualWrite(V15, ((param_hora3 * 60 + param_minuto3) * 60), stopAt, tz); //Tempo(segundos) = Hora*60*60 + Minuto*60 + Segundos (terceiro horário de irrigação)


  if (param_erro == 1) {

    Blynk.virtualWrite(V1, 255); //Envia para V1 -> 255 para acender totalmente o LED

  } else {

    Blynk.virtualWrite(V1, 0); //Envia para V1 -> 0 para apagar totalmente o LED

  }


  if (param_modo_irriga == 1) {

    Blynk.virtualWrite(V8, 1); //Envia para V1 -> 255 para acender totalmente o LED

  } else {

    Blynk.virtualWrite(V8, 2); //Envia para V1 -> 0 para apagar totalmente o LED

  }

  if (digitalRead(VALV) == LOW) {//acende e e apaga o LED de irrigação

    Blynk.virtualWrite(V11, 255); //Envia para V1 -> 255 para acender totalmente o LED

  } else {
```

```
    Blynk.virtualWrite(V11, 0); //Envia para V1 -> 0 para apagar totalmente o LED
  }

}//FIM void myTimerEvent() {
// L170 --- FUNÇÕES DE USO DO APLICATIVO BLYNK

// --- APLICATIVO V0 --- OBS: Define hora e minutos 1 da irrigação
BLYNK_WRITE(V0) {
  TimeInputParam t(param);
  // Process start time
  if (t.hasStartTime())
  {
    //Serial.println(String("Start: ") + t.getStartHour() + ":" + t.getStartMinute() + ":" + t.getStartSecond());
    if (param_hora1 != t.getStartHour()) {
      param_hora1 = t.getStartHour();
      EEPROM.write(HORADEIRRIGAR, param_hora1);
      delay(50);
    }
    if (param_minuto1 != t.getStartMinute()) {
      param_minuto1 = t.getStartMinute();
      EEPROM.write(MINUTODEIRRIGAR, param_minuto1);
      delay(50);
    }
    if (EEPROM.commit()) {
      Serial.println("L190 EEPROM Irrigacao 1: " + String(param_hora1) + 'h' + String(param_minuto1) + "min gravada com sucesso!");
```

```
    terminal.println("L190 EEPROM Irrigacao 1: " + String(param_hora1) + 'h' +
String(param_minuto1) + "min gravada com sucesso!");
    terminal.flush();
   } else {
    Serial.println("L190 ERROR! EEPROM Hora/Minuto 1 não gravada!");
    terminal.println("L190 ERROR! EEPROM Hora/Minuto 1 não gravada!");
    terminal.flush();
   }
 }
 else if (t.isStartSunrise()) {
  Serial.println("L200 Inicio ao amanhecer!");
 }
 else if (t.isStartSunset()) {
  Serial.println("L200 Inicio ao entardecer!");
 }
 else {
  // Do nothing
 }
 //==================================
 // Process stop time
 if (t.hasStopTime()) {
  Serial.println(String("L210 Stop: ") +
                 t.getStopHour() + ":" +
                 t.getStopMinute() + ":" +
                 t.getStopSecond());
 }
 else if (t.isStopSunrise()) {
  Serial.println("L210 Fim ao amanhecer!");
```

```
  } else if (t.isStopSunset()) {
    Serial.println("L210 Fim ao entardecer!");
  } else {
    // Do nothing: no stop time was set
  }

  // Process timezone// Timezone is already added to start/stop time
  Serial.println(String("L220 Time zone: ") + t.getTZ() + String("\nTime zone offset: ") + t.getTZ_Offset());
  // Get timezone offset (in seconds)

  // Process weekdays (1. Mon, 2. Tue, 3. Wed, ...)
  //  for (int i = 1; i <= 7; i++) {
  //    if (t.isWeekdaySelected(i)) {
  //      Serial.println(String("Day ") + i + " is selected");
  //    }
  //  }
  //Serial.println();
}//FIM BLYNK_WRITE(V0) {
// --- APLICATIVO V2 --- OBS: Define IRRADIAÇÃO a ser irrigada
BLYNK_WRITE(V2) {
  float valorLido = param.asFloat();
  //float valorLido = param.asDouble();

  if (param_irradiacao != valorLido) {
    param_irradiacao = valorLido;
  }
}//FIM BLYNK_WRITE(V2){
```

```
// --- APLICATIVO V3 --- OBS: Define Fator DELTA do calculo da irrigação
BLYNK_WRITE(V3) {
  float valorLido = param.asFloat();
  //float valorLido = param.asDouble();

  if (param_delta != valorLido) {
    param_delta = valorLido;
  }
}//FIM BLYNK_WRITE(V3){
// --- APLICATIVO V7 --- OBS: comandos vindo da serial
BLYNK_WRITE(V7) {
  String  valorLido = param.asStr();
  if (valorLido == "h") {
    //--- cria uma string MENU que escreve um menu na tela do TV
    char f_menu [] =  "0-Informações [info]\n"
                      "1-Temperatura [temp]\n"
                      "2-Umidade     [umid]\n"
                      "3-Endereço IP [ ip ]\n"
                      "4-Rede SSID   [ssid]\n"
                      "h-Ajuda       [ h ]\n"
                      "c-Limpar tela [clear]\n";
    Serial.println(f_menu);
    //terminal.clear();
    terminal.println(f_menu);
  }
  else if (valorLido == "ip") {
    Serial.print("IP: ");
```

```
    Serial.println(WiFi.localIP());
    terminal.print("IP: ");
    terminal.println(WiFi.localIP());
  }
  else if (valorLido == "ssid") {
    Serial.println("SSID: " + String(ssid));
    terminal.println("SSID: " + String(ssid));
  }
  else if (valorLido == "umid") {
    Serial.println("Umin (atual/min/máx): " + String(param_umid) + '/' + String(umid_min) + '/'
+ String(umid_max));
    terminal.println("Umin (atual/min/máx): " + String(param_umid) + '/' + String(umid_min) +
'/' + String(umid_max));
  }
  else if (valorLido == "temp") {
    Serial.println("Temp (atual/min/máx): " + String(param_temp) + '/' + String(temp_min) + '/'
+ String(temp_max));
    terminal.println("Temp (atual/min/máx): " + String(param_temp) + '/' + String(temp_min) +
'/' + String(temp_max));
  }
  else if (valorLido == "clear") {
    Serial.println("Limpando tela terminal virtual");
    terminal.clear();
  }
  else if (valorLido == "info") {
    Serial.println(dataString);
    terminal.println(dataString);
  }
  else if (valorLido == "calc") {
```

```
    Serial.println(inicioirrigaString);
    terminal.println(inicioirrigaString);
  }
  else {
    Serial.print("Comando não reconhecido! Envie 'h' para ajuda" );
    terminal.print("Comando não reconhecido! Envie 'h' para ajuda");

  }
  // Assegurar que dados foram enviados
  terminal.flush();
}//FIM BLYNK_WRITE(V7){

// --- APLICATIVO V8 --- OBS: Define modo manual ou automático
BLYNK_WRITE(V8) {
  switch (param.asInt()) {
    case 1: { // MANUAL
        param_modo_irriga = 1; //1 = modo manual
        Serial.println("Modo manual ativado!");
        terminal.println("Modo manual ativado!");
        terminal.flush();
        break;
      }
    case 2: { // AUTOMÁTICO
        param_modo_irriga = 0;//0 = modo automático
        param_irrigar_manual = 0; //desliga irrigação manual se estiver ligada
        Blynk.virtualWrite(V9, param_irrigar_manual);//Envia para V9 -> atualizar os dados
        digitalWrite(VALV, HIGH);  //desliga valvula de irrigação
```

```
      Serial.println("Modo automatico ativado!");
      terminal.println("Modo automatico ativado!");
      terminal.flush();
      break;
    }
  }
}//FIM BLYNK_WRITE(V8){
// --- APLICATIVO V9 --- OBS: Define Fator IRRIGAÇÃO MANUAL ligada ou desligada
BLYNK_WRITE(V9) {
  bool valorLido = param.asInt();
  if (param_modo_irriga == 1) {
    if (valorLido == 1) {
      param_irrigar_manual = 1; //inicia irrigação manual
      digitalWrite(VALV, LOW);  //liga valvula de irrigação
      cont_ti_manual = 0;       //contador de tempo de irrigação
      Serial.println("Irrigacao manual iniciada");
      terminal.println("Irrigacao manual iniciada");
      terminal.flush();
    }
    else {
      param_irrigar_manual = 0;
      digitalWrite(VALV, HIGH);  //desliga valvula de irrigação
      Serial.println("Irrigacao manual finalizada");
      terminal.println("Irrigacao manual finalizada");
      terminal.flush();
    }
  } else {
```

```
    Blynk.virtualWrite(V9, 0);//Envia para V9 -> atualizar os dados
  }
}//FIM BLYNK_WRITE(V9){
// --- APLICATIVO V10 --- OBS: Define Fator TI manual
BLYNK_WRITE(V10) {
  byte valorLido = param.asInt();
  if (tempo_ti_manual != valorLido) {
    tempo_ti_manual = valorLido;
    Serial.println("TI manual ajustado: " + String(tempo_ti_manual) + "min");
    terminal.println("TI manual ajustado: " + String(tempo_ti_manual) + "min");
    terminal.flush();
  }//FIM if (tempo_ti_manual != valorLido) {
}//FIM BLYNK_WRITE(V10){

// --- APLICATIVO V12 --- OBS: Envia para V12 o valor do numero de irrigações
BLYNK_WRITE(V12) {
  byte valorLido = param.asInt();
  if (param_Nirriga != valorLido) {
    param_Nirriga = valorLido;

    if (param_Nirriga < 1) {
      Serial.print("L420 Numero de irrigacoes menor que 1: ");
      Serial.println(param_Nirriga);
      terminal.print("L420 Numero de irrigacoes menor que 1: ");
      terminal.println(param_Nirriga);
      terminal.flush();
      param_Nirriga = 1;
```

```
  }
  if (param_Nirriga > 3) {
   Serial.print("L420 Numero de irrigacoes maior que 3: ");
   Serial.println(param_Nirriga);
   terminal.print("L420 Numero de irrigacoes maior que 3: ");
   terminal.println(param_Nirriga);
   terminal.flush();
   param_Nirriga = 3;
  }

  EEPROM.write(NIRRIGA, param_Nirriga);
  delay(50);

  if (EEPROM.commit()) {
   Serial.print("L420 Numero de irrigacoes atualizado: ");
   Serial.println(param_Nirriga);
   terminal.print("L420 Numero de irrigacoes atualizado: ");
   terminal.println(param_Nirriga);
   terminal.flush();
  } else {
   Serial.println("L1430 ERROR! EEPROM NIRRIGA não gravada!");
   terminal.println("L430 ERROR! EEPROM NIRRIGA não gravada!");
   terminal.flush();
  }
 }//FIM if (param_Nirriga != valorLido) {
}//FIM BLYNK_WRITE(V12){
```

```
//
================================================================================
================================================================================
======================

// --- APLICATIVO V14 --- OBS: Define hora e minutos 2 da irrigação

BLYNK_WRITE(V14) {

  TimeInputParam t(param);

  // Process start time

  if (t.hasStartTime())

  {

    //Serial.println(String("Start: ") + t.getStartHour() + ":" + t.getStartMinute() + ":" +
t.getStartSecond());

    if (param_hora2 != t.getStartHour()) {

      param_hora2 = t.getStartHour();

      EEPROM.write(HORADEIRRIGAR2, param_hora2);

      delay(50);

    }

    if (param_minuto2 != t.getStartMinute()) {

      param_minuto2 = t.getStartMinute();

      EEPROM.write(MINUTODEIRRIGAR2, param_minuto2);

      delay(50);

    }

    if (EEPROM.commit()) {

      Serial.println("L190 EEPROM Irrigacao 2: " + String(param_hora2) + 'h' +
String(param_minuto2) + "min gravada com sucesso!");

      terminal.println("L190 EEPROM Irrigacao 2: " + String(param_hora2) + 'h' +
String(param_minuto2) + "min gravada com sucesso!");

      terminal.flush();

    } else {
```

```
    Serial.println("L190 ERROR! EEPROM Hora2/Minuto2 não gravada!");
    terminal.println("L190 ERROR! EEPROM Hora2/Minuto2 não gravada!");
    terminal.flush();
  }
}
else if (t.isStartSunrise()) {
  Serial.println("L200 Inicio ao amanhecer!");
}
else if (t.isStartSunset()) {
  Serial.println("L200 Inicio ao entardecer!");
}
else {
  // Do nothing
}
//================================
// Process stop time
if (t.hasStopTime()) {
  Serial.println(String("L210 Stop: ") +
                 t.getStopHour() + ":" +
                 t.getStopMinute() + ":" +
                 t.getStopSecond());
}
else if (t.isStopSunrise()) {
  Serial.println("L210 Fim ao amanhecer!");
} else if (t.isStopSunset()) {
  Serial.println("L210 Fim ao entardecer!");
} else {
```

```
    // Do nothing: no stop time was set
  }

  // Process timezone// Timezone is already added to start/stop time
  Serial.println(String("L220 Time zone: ") + t.getTZ() + String("\nTime zone offset: ") + t.getTZ_Offset());
  // Get timezone offset (in seconds)

  // Process weekdays (1. Mon, 2. Tue, 3. Wed, ...)
  //  for (int i = 1; i <= 7; i++) {
  //    if (t.isWeekdaySelected(i)) {
  //      Serial.println(String("Day ") + i + " is selected");
  //    }
  //  }
  //Serial.println();
}//FIM BLYNK_WRITE(V14) {

// --- APLICATIVO V15 --- OBS: Define hora e minutos 3 da irrigação
BLYNK_WRITE(V15) {
  TimeInputParam t(param);
  // Process start time
  if (t.hasStartTime())
  {
    //Serial.println(String("Start: ") + t.getStartHour() + ":" + t.getStartMinute() + ":" + t.getStartSecond());
    if (param_hora3 != t.getStartHour()) {
      param_hora3 = t.getStartHour();
      EEPROM.write(HORADEIRRIGAR3, param_hora3);
```

```
    delay(50);
   }
   if (param_minuto3 != t.getStartMinute()) {
    param_minuto3 = t.getStartMinute();
    EEPROM.write(MINUTODEIRRIGAR3, param_minuto3);
    delay(50);
   }
   if (EEPROM.commit()) {
    Serial.println("L190 EEPROM Irrigacao 3: " + String(param_hora2) + 'h' +
String(param_minuto2) + "min gravada com sucesso!");
    terminal.println("L190 EEPROM Irrigacao 3: " + String(param_hora2) + 'h' +
String(param_minuto2) + "min gravada com sucesso!");
    terminal.flush();
   } else {
    Serial.println("L190 ERROR! EEPROM Hora3/Minuto3 não gravada!");
    terminal.println("L190 ERROR! EEPROM Hora3/Minuto3 não gravada!");
    terminal.flush();
   }
  }
  else if (t.isStartSunrise()) {
   Serial.println("L200 Inicio ao amanhecer!");
  }
  else if (t.isStartSunset()) {
   Serial.println("L200 Inicio ao entardecer!");
  }
  else {
   // Do nothing
  }
```

```
//================================

// Process stop time

if (t.hasStopTime()) {

  Serial.println(String("L210 Stop: ") +

                 t.getStopHour() + ":" +

                 t.getStopMinute() + ":" +

                 t.getStopSecond());

}

else if (t.isStopSunrise()) {

  Serial.println("L210 Fim ao amanhecer!");

} else if (t.isStopSunset()) {

  Serial.println("L210 Fim ao entardecer!");

} else {

  // Do nothing: no stop time was set

}


// Process timezone// Timezone is already added to start/stop time

Serial.println(String("L220 Time zone: ") + t.getTZ() + String("\nTime zone offset: ") + t.getTZ_Offset());

// Get timezone offset (in seconds)


// Process weekdays (1. Mon, 2. Tue, 3. Wed, ...)

//  for (int i = 1; i <= 7; i++) {

//    if (t.isWeekdaySelected(i)) {

//      Serial.println(String("Day ") + i + " is selected");

//    }

//  }

//Serial.println();
```

```
}//FIM BLYNK_WRITE(V15) {
// --- APLICATIVO V16 --- OBS: Comando manual de leitura de sensores
BLYNK_WRITE(V16) {
  byte valorLido = param.asInt();
  if (valorLido == 1) {
    LerSensor_DHT11();
    Serial.print("L570 Leitura manual dos sensores: ");
    Serial.print(param_temp);
    Serial.print("º  /  ");
    Serial.print(param_umid);
    Serial.println("%");
    terminal.print("L570 Leitura manual dos sensores: ");
    terminal.print(param_temp);
    terminal.print("º  /  ");
    terminal.print(param_umid);
    terminal.println("%");
    terminal.flush();
  }//FIM if (valorLido == 1) {
}//FIM BLYNK_WRITE(V16){
// L330 --- void setup ---
void setup()
{
  // BEGIN necessários
  Serial.begin(9600);
  EEPROM.begin(512);
  delay(2000); // wait for console opening
```

```
// --- estabelece inicio do sistema
Serial.println("\n\nL350 IA Secundário vIA2 v0.4.4.3 inicializado...");

// --- definindo portas e sinalizações de alerta
pinMode(VALV, OUTPUT); //definindo botão de entrada //no circuito está PULLDOWN
digitalWrite(VALV, HIGH);  //desliga valvula de irrigação

pinMode(RUN, OUTPUT); //definindo botão de entrada //no circuito está PULLDOWN
digitalWrite(RUN, HIGH);  //desliga valvula de irrigação

pinMode(ERROSD, OUTPUT); //definindo botão de entrada //no circuito está PULLDOWN
digitalWrite(ERROSD, LOW);  //desliga valvula de irrigação

pinMode(WM_RESET, INPUT); //botão para reconfigurar WiFi (WM) //endereço de acesso 10.0.0.1
////--- Define VALORES em EEPROM e em suas respectivas variáveis
//                EEPROM.write(HORADEIRRIGAR, 12); delay(50); EEPROM.write(MINUTODEIRRIGAR, 23);
//   delay(50);
//   EEPROM.put(IRRADIACAO, 12.2f);    delay(50);  EEPROM.put(FATORDELTA, 1.2f);
//   delay(50);
//
//   if (EEPROM.commit()) {
//     Serial.println("EEPROM gravada com sucesso!");
//     terminal.println("EEPROM gravada com sucesso!");
//   } else {
//     Serial.println("ERROR! EEPROM não gravada!");
//     terminal.println("ERROR! EEPROM não gravada!");
```

```
//    }


 param_hora1 = EEPROM.read(HORADEIRRIGAR); param_minuto1 = EEPROM.read(MINUTODEIRRIGAR);

 param_hora2 = EEPROM.read(HORADEIRRIGAR2); param_minuto2 = EEPROM.read(MINUTODEIRRIGAR2);

 param_hora3 = EEPROM.read(HORADEIRRIGAR3); param_minuto3 = EEPROM.read(MINUTODEIRRIGAR3);

 param_Nirriga = EEPROM.read(NIRRIGA);


 EEPROM.get(IRRADIACAO, param_irradiacao); EEPROM.get(FATORDELTA, param_delta); //pega valor float de FATORDELTA e joga em param_delta


 //  Serial.println("Valores das variáveis EEPROM");

 //  Serial.println(param_hora1);       Serial.println(param_minuto1);

 //  Serial.println(param_irradiacao); Serial.println(param_delta);

 // --- WIFI MANAGER

 config_WM(); //configura o WiFi Manager //final do sketch

 // --- CONEXÃO WIFI

 Serial.print("L390 Conectando com: ");

 Serial.println(ssid);


 WiFi.begin(ssid, pass);

 while (WiFi.status() != WL_CONNECTED) {

   delay(500);

   Serial.print(".");

   //Serial.print(millis());

   if (millis() > 15000) { //espera 15 segundos tentando conectar, caso não consiga, continua, mesmo offline
```

```
    Serial.println("L400 Não conectado ao Wifi...continuando o programa offline!");
    break;
   }
  }
  // --- CONEXÃO BLYNK
  if (WiFi.status() == WL_CONNECTED) {
   Serial.print("\nL400 Dispositivo online no IP: ");
   Serial.println(WiFi.localIP());

   Blynk.config(auth);//inicializa servidor blynk
   // --- CONEXÃO NTP UDP
   Serial.println("L410 Conectando a rtc NTP...");
   Udp.begin(localPort);
   setSyncProvider(getNtpTime);
   setSyncInterval(30 * 60); // tempo de sincronização em segundos (ideal a cada 10 ou 20 minutos)

   if (getNtpTime != 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso seja zero, repete no próximo ciclo.
    param_reconexao = 1;
   } else {
    param_reconexao = 0;
   }
  }// FIM if (WiFi.status() == WL_CONNECTED) {
  // --- INICIALIZA DHT
  Serial.print("L330 Inicializando DHT...");
  dht.begin();
  Serial.println("L330 DHT inicializado com sucesso!");
```

```
  // --- VERIFICA SD //faz primeira gravação de inicialização
  Serial.println("L450 Inicializando  SD...");
  dataString = "L450 Iniciando sistema...começando leituras";
  gravarSD(dataString, "LOG.txt");

  if (param_erro == 1) {
    digitalWrite(ERROSD, LOW);  //acende LED se houver erro na gravação SD //atua no pino diretamente, caso haja erro até início do Loop
  } else {
    digitalWrite(ERROSD, HIGH);  //apaga LED se não houver erro na gravação SD //atua no pino diretamente, caso haja erro até início do Loop
  }

  //=======================================
  // --- OTA ---
  config_OTA(); //função pra configurar OTA //localizada no final do sketch

  //=======================================
  // --- ENVIO CONSTANTE DE INFORMAÇÕES PARA O BLYNK
  timer.setInterval(3500L, myTimerEvent); //atualização a cada 5000L = 5 segundos

}//FIM setup()

//Variáveis para Loop
time_t prevDisplay = 0; //cria variável para controle de tempo
//WiFiManager wifiManager;//cria variável de controle no WiFi Mananger (WM)
```

```
// L480 --- void loop ---

void loop()

{

  //---CICLOS INSTANTANEOS

  Blynk.run(); // Faz inicialização do Blynk

  timer.run(); // Initiates BlynkTimer

  chama_WM(WM_RESET);

  ArduinoOTA.handle();


  //---CICLOS DE 1 SEGUNDO


  if (now() != prevDisplay) { //atualiza valor mostrado se valor mudar //Atualiza prevDisplay e gera ciclos de 1 segundo e faz piscar LED de controle


    prevDisplay = now();      //prevDisplay recebe o valor atual do tempo now()

    verif_segundo = second();


    if (param_irrigar_manual == 1) {//se irrigação manual estiver ativada, faz alteração no contador de irrigação

      Serial.println("Cont_ti_manual");

      Serial.println(cont_ti_manual);

      cont_ti_manual++; //conta tempo de ti manual

    }

    if (param_irrigar_auto == 1) {//se irrigação automática estiver ativada, faz alteração no contador de irrigação

      Serial.println("Cont_ti_auto");

      Serial.println(cont_ti_auto);

      cont_ti_auto++; //conta tempo de ti auto
```

```
  }


  //---Pisca LEDs --- RUN, WiFi e SD
  if (WiFi.status() != WL_CONNECTED) { //se conectado pisca LED RUN, se desconectado, pisca LED por tempo mais curto
    digitalWrite(RUN, 1); //liga LED RUN
    delay(100);
    digitalWrite(RUN, 0); //desliga LED RUN
    //Serial.println("L470 RUN ligado!");
  } else {
    if (param_run == 1) {
      digitalWrite(RUN, 0); //led desligado
      //Serial.println("L470 RUN desligado!");
    } else {
      digitalWrite(RUN, 1); //led ligado
      //Serial.println("L470 RUN ligado!");
    }
  }


  if ((param_erro == 1) && (param_run == 1)) {
    //if (param_erro == 1) {
    digitalWrite(ERROSD, 1);//liga LED de erro
    //Serial.println("L470 ERRO ligado!");
  } else {
    digitalWrite(ERROSD, 0);//desliga LED de erro
    //Serial.println("L470 ERRO desligado!");
  }
```

```
  if (param_run == 1) {
   param_run = 0;
  } else {
   param_run = 1;
  }



  //=======================================
  //--- Verifica se dados foram enviados a Blynk ou desconectado.
  if (Blynk.connected() != 0) { //verifica se Blynk está conectado
   if (verif_segundo % 30 == 0) { //a cada 30 segundos prepara dados a serem enviados
    if (param_inicial == 0) {
     terminal.println("L530 SISTEMA IA vIA2 v0.4.4.3 inicializado!"); terminal.flush();
//assegura se dados foram enviados com sucesso
     param_inicial = 1; //muda o parametro para informar que dado foi enviado
     param_reconBlynk = 1; //se inicializado não precisa enviar msg informando que conectou
blynk
    }
    if (param_reconBlynk == 0) {
     terminal.println("L530 [" + porzero_int(hour()) + "h" + porzero_int(minute()) + "min]
ALERTA DE RECONEXÃO!"); terminal.flush(); //assegura se dados foram enviados com
sucesso
     param_reconBlynk = 1; //muda o parametro para informar que dado foi enviado
    }
    if (param_tv_leitura == 1) {
     terminal.println(leituraString);
     terminal.flush(); //assegura se dados foram enviados com sucesso
     param_tv_leitura = 0; //muda o parametro para informar que dado foi enviado
    }
```

```
      if (param_tv_inicioirriga == 1) {
        terminal.println(inicioirrigaString); //envia string de inicio de irrigação
        terminal.flush(); //assegura se dados foram enviados com sucesso
        param_tv_inicioirriga = 0; //muda o parametro para informar que dado foi enviado
      }
      if (param_tv_fimirriga == 1) {
        terminal.println(fimirrigaString); //envia so os dados importantes
        terminal.flush(); //assegura se dados foram enviados com sucesso
        param_tv_fimirriga = 0; //muda o parametro para informar que dado foi enviado
      }

    }//FIM if (verif_segundo % 30 == 0)){
  }//FIM if ((Blynk.connected() != 0){
  else {
    param_reconBlynk = 0;
  }

  //=====================================
  //--- Verificar situação se houve desconexão e tenta reconetar a Blynk e rtc NTP
  if (verif_segundo % 30 == 0) {
    if ((WiFi.status() == WL_CONNECTED) && ((Blynk.connected() == 0) || (param_reconexao == 0))) //verifica a cada minuto se está conectado e blink conectado ou foi uma reconexão
    {
      //---se Blynk não conectado tenta conexão
      if (Blynk.connected() == 0) {
        Serial.println("L560 Conectando a BLYNK...");
```

```
        Blynk.config(auth);//inicializa servidor blynk
      }
      //    Serial.print("L530 Valor de Blynk.connected(): ");
      //    Serial.println(Blynk.connected());

      //---recebe pacote de rtc NTP e atualiza hora e data
      if (param_reconexao == 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso seja zero, repete no próximo ciclo.
        Serial.println("L570 Conectando a rtc NTP...");
        Udp.begin(localPort);
        setSyncProvider(getNtpTime);
        setSyncInterval(30 * 60); // tempo de sincronização em segundos (ideal a cada 10 ou 20 minutos)

        if (getNtpTime != 0) {  //se conseguiu conectar com NTP e Blynk não precisa repetir passo. Caso seja zero, repete no próximo ciclo.
          param_reconexao = 1;
        } else {
          param_reconexao = 0;
        }
      } //FIM if (param_reconexao == 0) {
    }//FIM if ((WiFi.status() == WL_CONNECTED)
  }//FIM if (second()%30 ==0){
 } //FIM if (now() != prevDisplay) OU if (cont_segundo != second())
 //FIM de CICLOS DE 1 SEGUNDO
 //---CICLOS DE 1 MINUTO

 //--- verifica se minuto mudou e atualiza
```

```
 if (verif_minuto != minute()) {
  verif_minuto = minute();

  //=======================================
  //--- Verificar situação da conexão
  if (WiFi.status() != WL_CONNECTED) {
   Serial.println("L540 Dispositivo desconectado, tentando conectar no WiFi...");
   param_reconexao = 0; //parametro para forçar reconexão com UDP e Blynk
   WiFi.begin(ssid, pass);
  }//FIM if (WiFi.status() != WL_CONNECTED) {


  //=======================================
  //--- Atualiza dados de EEPROM
  //---IRRADIAÇÃO
  EEPROM.get(IRRADIACAO, param_verifica);
  if (param_irradiacao != param_verifica) {
    EEPROM.put(IRRADIACAO, param_irradiacao); //usa-se EEPROM.put pois o falor inserido é float, se fosse byte, seria write
    delay(50);
    if (EEPROM.commit()) {
      EEPROM.get(IRRADIACAO, param_verifica);
      Serial.println("L600 EEPROM IRRADIAÇÃO gravada com sucesso: " + String(param_verifica, 2));
      terminal.println("L600 EEPROM IRRADIAÇÃO gravada com sucesso: " + String(param_verifica, 2));
    } else {
      Serial.println("L600 ERROR! EEPROM não gravada!");
```

```
      terminal.println("L600 ERROR! EEPROM não gravada!");
    }
    terminal.flush(); //assegura se dados foram enviados com sucesso
    //Blynk.virtualWrite(V2, param_irradiacao); //Envia para V2 o valor da irradiação //enviado junto com o resto
  }//FIM if (param_irradiacao != EEPROM.get(IRRADIACAO)) {

  //=======================================
  //---FATOR DELTA
  EEPROM.get(FATORDELTA, param_verifica);
  if (param_delta != param_verifica) {
    EEPROM.put(FATORDELTA, param_delta); //usa-se EEPROM.put pois o falor inserido é float, se fosse byte, seria write
    delay(50);
    if (EEPROM.commit()) {
      EEPROM.get(FATORDELTA, param_verifica);
      Serial.println("L620 EEPROM FATORDELTA gravada com sucesso: " + String(param_verifica));
      terminal.println("L620 EEPROM FATORDELTA gravada com sucesso: " + String(param_verifica));
    } else {
      Serial.println("L620 ERROR! EEPROM não gravada!");
      terminal.println("L620 ERROR! EEPROM não gravada!");
    }
    terminal.flush(); //assegura se dados foram enviados com sucesso
    //Blynk.virtualWrite(V3, param_delta); //Envia para V3 o valor de delta //enviado junto com o resto
```

```
  }//FIM if (param_delta != EEPROM.get(FATORDELTA)) {



  //=======================================

  //---Verifica a cada minuto se SD foi gravado corretamente, se não tenta gravar novamente.

  if (param_erro == 1) {

    gravarSD(dataString, "LOG.txt"); //

  }



} //FIM if (verif_minuto != minute()) {

//--- FIM de CICLOS DE 1 MIN

//--- CICLOS DE 1 HORA

// --- verifica se passou um determinado tempo(hora ou minuto), para fazer uma nova leitura de temperatura e umidade


if (verif_hora != hour())

  //if (verif_hora != minute()) //para testes

{

  verif_hora = hour();

  //verif_hora = minute();  //para testes



  //=======================================

  //faz a leitura da umidade e grava nas variáveis dht, se houver leitura nan, então tenta fazer 5 leituras para tentar uma nova leitura

  Serial.println("L920 Fazendo leitura de umidade e temperatura");
```

```
  LerSensor_DHT11(); //função para ler 5x o sensor e verificar valores e se forem valores NAN, descarta.


  //scrip antigo para leitura de sensor //agora usa função

  /*   byte i = 0; //faz contagem de processos

     do { //faz leitura de umidade e temperatura se valor recebido for NAM, faz nova leitura.

       dht_umid = dht.readHumidity();

       dht_temp = dht.readTemperature();

       i++;

       delay (500);

     } while ((isnan(dht_umid) || isnan(dht_temp)) && (i < 5)); //verifica se leitura dht é nam e se contador é menor que 5



     //=====================================

     // --- verificação de temperatura e umidade é valor válido ---

     // --- verifica se valor lido é valido, se sim adiciona no banco de dados, caso não, define zero!

     if (isnan(dht_temp) || isnan(dht_umid)) { //  se valor for nan,

       bd_temp[cont_bd] = 0;     // banco de dados recebe zero;

       param_temp = 0;

       bd_umid[cont_bd] = 0;

       param_umid = 0;

       Serial.println("Valores lidos pelo sensor INVÁLIDOS, definindo valor ZERO!");

       terminal.println("Valores lidos pelo sensor INVÁLIDOS, definindo valor ZERO!");

       terminal.flush();


     } else { // senão define zero
```

```
        bd_temp[cont_bd] = dht_temp;  //banco de dados recebe valor lido;
        param_temp = dht_temp;
        bd_umid[cont_bd] = dht_umid;
        param_umid = dht_umid;
        Serial.println("Valores lidos pelo sensor VÁLIDOS");
        terminal.println("Valores lidos pelo sensor VÁLIDOS");
        terminal.flush();
      }
   */

   //=======================================

   // MAX E MIN // ---Verifica que valores são máx ou mín no banco de dados para guardar---
   temp_max = -1;    //zerando valores para verificar quais valores do banco de dados é o maior e menor
   temp_min = 99;   //zerando valores para verificar quais valores do banco de dados é o maior e menor
   Serial.println("Verificando banco de dados");
   for (byte i = 0; i < 24; i++) {
     if ((bd_temp[i] > temp_max) && (bd_temp[i] != 0)) { //verifica se dado recebido é min ou max
       temp_max = bd_temp[i];
     }
     if ((bd_temp[i] < temp_min) && (bd_temp[i] != 0)) {
       temp_min = bd_temp[i];
     }
     if ((bd_umid[i] > umid_max) && (bd_umid[i] != 0)) { //verifica se dado recebido é min ou max
       umid_max = bd_umid[i];
     }
```

```
  if ((bd_umid[i] < umid_min) && (bd_umid[i] != 0)) {
    umid_min = bd_umid[i];
  }
 }//FIM for (byte i = 0; i < 24; i++) {

 //  ======================================
 // --- MÉDIA DOS VALORES ---
 // --- faz a soma dos valores armazenados
 float soma_umid = 0; //armazena a soma de valores para tirar a média
 float soma_temp = 0;
 byte  soma_erro = 0;//armazena erros para se necessário abater da média

 for (byte i = 0; i < 24; i++) {
  if ((bd_temp[i] == 0) || (bd_umid[i] == 0)) {
    soma_erro++; //se valor de bd_tempo = 0; houve erro na leitura, então soma_erro aumenta um valor para ser abatido depois
  } else {
    soma_umid = soma_umid + bd_umid[i];
    soma_temp = soma_temp + bd_temp[i];
  }
 }//FIM for(int i = 0; i < 24; i++)

 // --- calcula a média desconsiderando os erros de leitura ---
 if (soma_erro < 24) { //se erros forem menor que o total, usa os valores válidos
  umid_media = soma_umid / (24 - soma_erro);
  temp_media = soma_temp / (24 - soma_erro);
 }
 else { //se os erros forem iguais aos valores totais, faz o valor da média igual a ZERO
```

```
    umid_media = 0;
    temp_media = 0;
  }
  //====================================
  // --- Gravando dados em String     // --- Gravação de String de dados no SD a cada hora
  dataString = escreve_datastring();
  if (WiFi.status() != WL_CONNECTED) { //se wifi não estiver conectado, alerta ao usuário, mas continua os trabalhos
    dataString += " [OFFLINE]";
  }

  gravarSD(dataString, "LOG.txt"); //imprime no SD a cada hora as leituras de hora em hora

  leituraString = "L1042 [" + porzero_int(year()) + "/" + porzero_int(month()) + "/" + porzero_int(day()) + "-[" + porzero_int(hour()) + "h" + porzero_int(minute()) + "min] " + porzero_float(param_temp, 1) + "ºC/" + porzero_float(param_umid, 1) + "%]"; //envia so os dados importantes
  param_tv_leitura = 1; //prepara para enviar para terminal virtual a string preparada

  cont_bd ++; //contador de unidades no banco de dados

  if (cont_bd > 23) {
    cont_bd = 0; //se banco de dados lotou, zera contador, para iniciar novo ciclo de gravação começando dos dados mais antigos
  }

 } //FIM if (verif_hora != hour())
 //FIM de CLICLO DE 1 HORA
 // --- CICLO DE IRRIGAÇÕES ---
```

```
  // --- Verifica fim do ciclo do dia, se prepara para o próximo

  //--- IRRIGAÇÃO MANUAL --- Inicio irrigação e fim irrigação

  if (param_modo_irriga == 1)//irrigação manual ativada

  { //Serial.println("Fim da irrigação manual!");

    if (cont_ti_manual > tempo_ti_manual * 60) { //cont_ti_manual*60 (contador está em segundos) //atualizado em: if (now() != prevDisplay)

      if (digitalRead(VALV) == LOW) {//se valvula estiver ligada, desliga

        digitalWrite(VALV, HIGH);  //desliga valvula de irrigação

        param_irrigar_manual = 0;

        cont_ti_manual = 0;

        Serial.println("Fim da irrigação manual!");

        terminal.println("Fim da irrigação manual!");

        terminal.flush();

      }

    }

  } //FIM  if (param_irrigar == true)

  //--- IRRIGAÇÃO AUTOMÁTICA --- Inicio irrigação e fim irrigação

  // else >> (param_modo_irriga == 0) ou seja, automático, faz o que se segue

  else if (param_irrigar_auto == 1) //quando o valor é atingido, zera parametros para realizar um novo ciclo de irrigação

  {

    if (digitalRead(VALV) == HIGH) {//verifica se valvula está desligada, se estiver, liga!

      digitalWrite(VALV, LOW);  //liga valvula de irrigação

      inicioirrigaString = "L820 IRRIGANDO! [Tmin/max/med/del/Fdel/Irrad]:\n";

      inicioirrigaString += "L820 [" + porzero_int(hour()) + 'h' + porzero_int(minute()) + "min] " + porzero_float(temp_min, 1) + '/' + porzero_float(temp_max, 1) + '/' + porzero_float(temp_media, 1) + '/' + porzero_float(temp_delta, 1) + '/' + String(param_delta, 2) + '/' + porzero_float(param_irradiacao, 1);

      Serial.println(inicioirrigaString);
```

```
    param_tv_inicioirriga = 1; //dispara flag para enviar msg para TV;
   }
   if (cont_ti_auto > tempo_ti_parcial) { //atualizado em: if (now() != prevDisplay)
    if (digitalRead(VALV) == LOW) {
      digitalWrite(VALV, HIGH);  //desliga valvula de irrigação
      param_irrigar_auto = 0;
      cont_ti_auto = 0;
      fimirrigaString = "L640 [" + porzero_int(hour()) + "h" + porzero_int(minute()) + "min] FIM da irrigação! TI:" + porzero_int(tempo_ti_parcial / 60) + "min" + porzero_int(tempo_ti_parcial % 60) + "seg"; //(tempo_ti_parcial%10)*60/10 para mostrar os segundos
      Serial.println(fimirrigaString);
      param_tv_fimirriga = 1; //parametro para realizar envio de dados para TV
    }
   }
  } //FIM  if (param_irrigar_auto == 1)
  //=====================================
  // --- MOMENTO DO CÁLCULO DE IRRIGAÇÃO E 1ª IRRIGAÇÃO

  if ((param_irrigar_auto == 0)
  &&(
  ((param_Nirriga >= 1) && (param_hora1 == hour()) && (param_minuto1 == minute())) ||
  ((param_Nirriga >= 2) && (param_hora2 == hour()) && (param_minuto2 == minute())) ||
  ((param_Nirriga == 3) && (param_hora3 == hour()) && (param_minuto3 == minute()))
  )
  ){ //chegou horário de irrigação, zera os valores para tirar a média do dia e iniciar proximo ciclo
```

// --- CÁLCULO DO TEMPO DE IRRIGAR //Modelo matemático pelo Método de Hargreaves-Samani

/*//Modelo matemático

// ETo = 0,0023*Qo*((Tmax-Tmin)^(0,5))*(Tmed+17,8) // ETc = ETo * Kc; // ETcLoc = ETc * KL; //

// ITN = ETcLoc/Ea; // Ti = (ITN*A)/(n*Qg);

// Assim juntando todas as funções: // Ti = 0,0023*Qo*((Tmax-Tmin)^(0,5))*(Tmed+17,8)*(Kc*KL*A)/(Ea*n*Qg)

// OU // Ti = 0,0023*((Tmax-Tmin)^(0,5))*(Tmed+17,8)*Qo*(Kc*KL*A)/(Ea*n*Qg)

// Qo = parametro irradiação

// Kc*KL*A/Ea*n*Qg = parametro delta

//(Kc*KL*A)/(Ea*n*Qg) onde:

ETo = Evapotranspiração basica

ETc = Evapotranspiração da cultura

Kc = Coeficiente da cultura

ETcLoc = Evapotranspiração localizada

ITN = Irrigação total necessária

Ti = tempo de irrigação

KL = coeficiente da localizada

Ea = Rendimento da irrigação = 0,85

A = Area do canteiro = 0,2*0,3 = 0,06

n = Número de irrigadores por planta = 1

Qg = Vazão do irrigador = 0,8 L/h

*/

```
  // --- se valor de diferença de temperatura menor que 5, estipula valor mínimo para evitar erros nos resultados
  if (temp_max - temp_min < 5) {
    temp_delta = 5;
  } else {
    temp_delta = temp_max - temp_min;
  }
  Serial.println("L1090 Temperatura max, min, delta e media: ");
  Serial.println(temp_max);
  Serial.println(temp_min);
  Serial.println(temp_delta);
  Serial.println(temp_media);
  terminal.println("L1090 Temperatura max, min, delta e media: ");
  terminal.println(temp_max);
  terminal.println(temp_min);
  terminal.println(temp_delta);
  terminal.println(temp_media);
  terminal.flush();


  // --- formula do tempo de irrigação TI em minutos // x60 para ajustar para minutos e x(60) para ajustar aos valores para segundos.
  tempo_ti_auto = 0.0023 * sqrt(temp_delta) * (temp_media + 17.8) * param_irradiacao * param_delta * 60 * 60; //tempo de irrigação em minutos número inteiro // *60*10 para ajustar valores
  if (param_Nirriga < 1) {
    param_Nirriga = 1;
  }
  if (param_Nirriga > 3) {
```

```
  param_Nirriga = 3;
}

tempo_ti_parcial = tempo_ti_auto / param_Nirriga;
Serial.println("L1100 Tempo calculado em segundos:");
Serial.println(tempo_ti_auto);
Serial.println(tempo_ti_parcial);
terminal.println("L1100 Tempo calculado em segundos:");
terminal.println(tempo_ti_auto);
terminal.println(tempo_ti_parcial);
terminal.flush();
//tempo_ti_auto = 99;

//=====================================
// --- Gravação dos dados diarios no dataString e gravação no SD.
dataString = escreve_datastring_diario();
gravarSD(dataString, "LOG.txt"); //parametros, 1o = String que será gravada no SD, 2o nome do arquivo que será gravado.
// --- INICIANDO NOVO CICLO DE IRRIGAÇÃO E LEITURAS
// L870 --- zerando variáveis // --- inicia-se processo de irrigação
cont_ti_auto = 0;       //zera contador, para início de contagem
param_irrigar_auto = 1; //liga parametro de controle de irrigação //autoriza a irrigação

// --- INICIANDO NOVO CICLO DE IRRIGAÇÃO PARCIAL
// L870 --- zerando variáveis // --- inicia-se processo de irrigação
Serial.println("Dando início a irragação da parcela diária");
terminal.println("Dando início a irragação da parcela diária");
terminal.flush();
```

```
  }//FIM if ((param_irrigar_auto == 0) && (param_hora1 >= hour()) && (param_minuto1 >= minute())) {


}//FIM void loop


// L850 --- void gravarSD(String f_texto, String f_arquivo) ---
void gravarSD(String f_texto, String f_arquivo)
{ //=======================================
  // --- Gravação de dados no SD
  if (!SD.begin(CS_PIN)) {
    param_erro = 1;
    Serial.println("L850 Falha, verifique se o cartão está presente!");
    terminal.println("L850 Falha, verifique se o cartão está presente!");
    terminal.flush(); //assegura se dados foram enviados com sucesso
    //ativa parametro de falha no SD para ligar LED se houver erro na gravação SD
    //digitalWrite(ERROSD, LOW); //acende LED se houver erro na gravação SD
  }
  else {
    Serial.println("L860 SD inicializado com sucesso!");
    File dataFile = SD.open(f_arquivo, FILE_WRITE);


    // se o arquivo foi aberto corretamente, escreve os dados nele
    if (dataFile) {
      //formatação no arquivo: linha a linha
      dataFile.println(f_texto);
      //fecha o arquivo após usá-lo
      dataFile.close();
```

```
      Serial.println("L870 O arquivo foi escrito com sucesso!");

      param_erro = 0; //desativa parametro de falha no SD para desligar LED de erro na gravação
SD

    }//FIM if (dataFile) {


    // se o arquivo não pôde ser aberto os dados não serão gravados.

    else {

      Serial.print("L880 Falha ao abrir o arquivo ");

      Serial.println(f_arquivo);

      terminal.print("L880 Falha ao abrir o arquivo " + String(f_arquivo));

      terminal.flush(); //assegura se dados foram enviados com sucesso

      param_erro = 1; //ativa parametro de falha no SD para ligar LED se houver erro na gravação
SD

    }//FIM else if (dataFile) {

  }//FIM else if (!SD.begin(CS_PIN)) {


  Serial.println(f_texto);

  //terminal.println(f_texto);


//terminal.println(String(porzero_int(year()))+"/"+String(porzero_int(month()))+"/"+String(po
rzero_int(day()))+"-
"+String(porzero_int(hour()))+"h"+String(porzero_int(minute()))+"min["+String(param_temp
)+"ºC]");

  //return;


}//FIM void gravarSD(String f_texto, String f_arquivo)

// --- String porzero_int(int num_x) --- //*OBS: serve para transformar valor em numero de 2
digitos para jogar na data e hora

String porzero_int(int num_x)

{ String novo_num_x = "";
```

```
  if (num_x < 10)

    novo_num_x += '0';

  novo_num_x += String(num_x);

  return novo_num_x;

}//FIM String porzero_int(int num_x)


// --- String porzero_float(float num_x, num_y) --- //*OBS: serve para transformar valor em numero de 2 digitos para jogar na data e hora

String porzero_float(float num_x, int num_y) //num_x é a entrada //num_y é o número de casas que deve aparecer

{ String novo_num_x = "";

  if (num_x < 10)

    novo_num_x += '0';

  novo_num_x += String(num_x, num_y);

  return novo_num_x;

}//FIM String porzero_float(float num_x, int num_y)

// --- String escreve_datastring(int num_x) --- //*OBS: serve para jogar os dados a serem escritos na serial e no SD //*OBS2: precisa da função "porzero_int();"

String escreve_datastring() {

  String str_x = "";

  str_x = ""; //limpa a variável de escrita de SD para iniciar um novo ciclo

  str_x += porzero_int(year());

  str_x += "/";

  str_x += porzero_int(month());

  str_x += "/";

  str_x += porzero_int(day());

  str_x += "-";

  str_x += porzero_int(hour());
```

```
  str_x += "h";
  str_x += porzero_int(minute());
  str_x += "min[";
  for (byte cont_i = 0; cont_i < 24; cont_i++) {
    str_x += porzero_float(bd_temp[cont_i], 1);
    //str_x += bd_temp[cont_i];
    if (cont_i < 23) str_x += "/"; //enquanto não chega no ultimo valor, escreve o separador
  }
  str_x += "]";
  return str_x;
}
//FIM String escreve_datastring() {
// --- String porzero_int(int num_x) --- //*OBS: serve para jogar os dados a serem escritos na serial e no SD //*OBS2: precisa da função "porzero_int();"
String escreve_datastring_diario() {
  String str_x = ""; //limpa a variável de escrita de SD para iniciar um novo ciclo
  str_x = "Leituras diárias Dia/TI/Umid/Temp(min,max,med): ";
  str_x += porzero_int(day());
  str_x += "/";
  str_x += porzero_int(month());
  str_x += "/";
  str_x += porzero_int(year());
  str_x += " - ";
  str_x += String(tempo_ti_auto / 60, 1); //tranforma em segundos
  str_x += "min [";
  str_x += umid_min;
  str_x += "/";
  str_x += umid_max;
```

```
  str_x += "/";
  str_x += umid_media;
  str_x += "|%][/[";
  str_x += temp_min;
  str_x += "/";
  str_x += temp_max;
  str_x += "/";
  str_x += temp_media;
  str_x += "|ºC]";
  return str_x;
}
//FIM String escreve_datastring_diario()

// --- Códigos para funcionamento do RTC NTP
const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

time_t getNtpTime()
{
  IPAddress ntpServerIP; // NTP server's ip address

  while (Udp.parsePacket() > 0) ; // discard any previously received packets
  Serial.println("L1000 Enviando solicitação a NTP...");
  // get a random server from the pool
  WiFi.hostByName(ntpServerName, ntpServerIP);
  Serial.print(ntpServerName);
  Serial.print(": ");
```

```
  Serial.println(ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("L1000 Resposta NTP recebida");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }

  Serial.println("L1020 SEM RESPOSTA DE NTP :-(");
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
```

```
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
//FIM time_t getNtpTime()
// --- WIFI MANAGER
// --- CONFIGURAÇÃO WIFI MANAGER
void config_WM() {
  // *obs: só precisa dessas funções pois não vai acionar o WM no início, apenas quando for apertado o botão
  //declaração do objeto wifiManager
  //WiFiManager wifiManager;

  //utilizando esse comando, as configurações são apagadas da memória
  //caso tiver salvo alguma rede para conectar automaticamente, ela é apagada.
```

```
  //wifiManager.resetSettings();


  //callback para quando entra em modo de configuração AP

  wifiManager.setAPCallback(configModeCallback);

  //callback para quando se conecta em uma rede, ou seja, quando passa a trabalhar em modo estação

  wifiManager.setSaveConfigCallback(saveConfigCallback);


}


// --- void chama_WM(int WM_pin) //Códigos para funcionamento do WiFi Manager (WM)

//=======================================================

void chama_WM(int WM_pin) {

  //gera a cada loop uma nova variável para armazenar os valores

  //WiFiManager wifiManager;  //gerava um envio demasiado de msg pra serial, pois a cada vez criado era enviada uma msg


  //se o botão foi pressionado

  if ( digitalRead(WM_pin) == 1 ) {

    //WiFiManager wifiManager;  //gerava um envio demasiado de msg pra serial, pois a cada vez criado era enviada uma msg

    unsigned long WM_tempodelay = millis();

    while ((digitalRead(WM_pin) == 1) && ((millis() - WM_tempodelay) < 5000) ) {

      Serial.print("Pino pressionado!");

      Serial.println(millis() - WM_tempodelay);

      delay(100);

    }
```

```
   if ((millis() - WM_tempodelay) > 5000) {
     Serial.println("Iniciando WiFi Manager!"); //tenta abrir o portal
     terminal.println("Iniciando WiFi Manager!"); //envia so os dados importantes para terminal virtual blynk
     terminal.flush(); //assegura se dados foram enviados com sucesso

     WiFi.disconnect();
     delay(1000);

     wifiManager.startConfigPortal("WiFi_Manager", "WEB123456789"); //entra no portal toda vez, ao inves de tentar se conectar a uma rede anterior.
     wifiManager.setMinimumSignalQuality(10); // % minima para ele mostrar no SCAN
     wifiManager.setRemoveDuplicateAPs(true); //remover redes duplicadas (SSID iguais)
     wifiManager.setConfigPortalTimeout(20); //timeout para o ESP nao ficar esperando para ser configurado para sempre
     wifiManager.setConnectTimeout(30);
   }
  }
}
//callback que indica que o ESP entrou no modo AP
void configModeCallback (WiFiManager * myWiFiManager)
{
  Serial.println("Entrou no modo de configuração");
  Serial.println(WiFi.softAPIP()); //imprime o IP do AP
  Serial.println(myWiFiManager->getConfigPortalSSID()); //imprime o SSID criado da rede
}

//callback que indica que salvamos uma nova rede para se conectar (modo estação)
```

```
void saveConfigCallback () {
  Serial.println("Configuração salva");
  Serial.println(WiFi.softAPIP()); //imprime o IP do AP
}

// --- OTA ---
//---CONFIGURAÇÃO OTA
void config_OTA() {
  // A porta fica default como 3232
  // ArduinoOTA.setPort(3232);

  // Define o hostname (opcional)
  ArduinoOTA.setHostname("TW_ESP8266_IA2");

  // Define a senha (opcional)
  ArduinoOTA.setPassword("WEB123456789");

  // É possível definir uma criptografia hash md5 para a senha usando a função "setPasswordHash"
  // Exemplo de MD5 para senha "admin" = 21232f297a57a5a743894a0e4a801fc3
  // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

  //define o que será executado quando o ArduinoOTA iniciar
  ArduinoOTA.onStart( startOTA ); //startOTA é uma função criada para simplificar o código

  //define o que será executado quando o ArduinoOTA terminar
  ArduinoOTA.onEnd( endOTA ); //endOTA é uma função criada para simplificar o código
```

```
  //define o que será executado quando o ArduinoOTA estiver gravando
  ArduinoOTA.onProgress( progressOTA ); //progressOTA é uma função criada para simplificar o código

  //define o que será executado quando o ArduinoOTA encontrar um erro
  ArduinoOTA.onError( errorOTA );//errorOTA é uma função criada para simplificar o código

  //inicializa ArduinoOTA
  ArduinoOTA.begin();
}

//funções de exibição dos estágios de upload (start, progress, end e error) do ArduinoOTA
//---MENSAGENS OTA
void startOTA()
{
  String type;

  //caso a atualização esteja sendo gravada na memória flash externa, então informa "flash"
  if (ArduinoOTA.getCommand() == U_FLASH)
    type = "flash";
  else   //caso a atualização seja feita pela memória interna (file system), então informa "filesystem"
    type = "filesystem"; // U_SPIFFS

  //exibe mensagem junto ao tipo de gravação
  Serial.println("Iniciando atualização " + type);
}
//---exibe mensagem fim
```

```
void endOTA()
{
  Serial.println("\nFIM");
}
//---exibe progresso em porcentagem
void progressOTA(unsigned int progress, unsigned int total)
{
  Serial.printf("Progresso: %u%%\r", (progress / (total / 100)));
}
//---caso aconteça algum erro, exibe especificamente o tipo do erro
void errorOTA(ota_error_t error)
{
  Serial.printf("Erro[%u]: ", error);

  if (error == OTA_AUTH_ERROR)
    Serial.println("Falha no inicio");
  else if (error == OTA_BEGIN_ERROR)
    Serial.println("Begin Failed");
  else if (error == OTA_CONNECT_ERROR)
    Serial.println("Falha na conexão");
  else if (error == OTA_RECEIVE_ERROR)
    Serial.println("Falha de recebimento");
  else if (error == OTA_END_ERROR)
    Serial.println("Falha na finalização");
}
// --- LER SENSOR DHT11
void LerSensor_DHT11() {
```

```
  byte i = 0; //faz contagem de processos
  do { //faz leitura de umidade e temperatura se valor recebido for NAM, faz nova leitura.
    dht_umid = dht.readHumidity();
    dht_temp = dht.readTemperature();
    i++;
    delay (500);
  } while ((isnan(dht_umid) || isnan(dht_temp)) && (i < 5)); //verifica se leitura dht é nam e se
contador é menor que 5



  //=====================================
  // --- verificação de temperatura e umidade é valor válido ---
  // --- verifica se valor lido é valido, se sim adiciona no banco de dados, caso não, define zero!
  if (isnan(dht_temp) || isnan(dht_umid)) { //  se valor for nan,
    bd_temp[cont_bd] = 0;     // banco de dados recebe zero;
    param_temp = 0;
    bd_umid[cont_bd] = 0;
    param_umid = 0;
    Serial.println("Valores lidos pelo sensor INVÁLIDOS, definindo valor ZERO!");
    terminal.println("Valores lidos pelo sensor INVÁLIDOS, definindo valor ZERO!");
    terminal.flush();

  } else { // senão define zero
    bd_temp[cont_bd] = dht_temp;  //banco de dados recebe valor lido;
    param_temp = dht_temp;
    bd_umid[cont_bd] = dht_umid;
    param_umid = dht_umid;
    Serial.println("Valores lidos pelo sensor VÁLIDOS");
```

```
    terminal.println("Valores lidos pelo sensor VÁLIDOS");
    terminal.flush();
  }

}

/*
  // --- INFORMAÇÕES IMPORTANTES

  //FUNCIONALIDADES:
  1 - Faz leituras horárias de temperatura e umidade e tira a média na mesma frequencia, faz calculo do TI para irrigar a cultura
  2 - Acesso e comandos pelo aplicativo Blynk e envio via terminal virtual
  3 - Atualização via OTA
  4 - Uso de WiFiManager para alterar a rede wifi utilizada

  //DEFINIÇÃO DE BOTÕES VIRTUAIS
  // V0 = HORA E MINUTO para irrigar (0~23h)
  // V1 = Erro de SD
  // V2 = Irradiação  // enviado para Blynk
  // V3 = Fator DELTA // enviado para Blynk  // junção dos fatores (Kc*KL*A)/(Ea*n*Qg) onde: Kc(coef da cultura) / KL(coef localizada) / A(área da planta) / Ea(rendimento da irrigação / n(numero de gotejadores por planta) / Qg(vazão por gotejador)
  // V4 = Temperatura lida de hora em hora   // para Superchart
  // V5 = Umidade lida de hora em hora       // para Superchart
  // V6 = Tempo de Irrigação
  // V7 = Terminal de informações
```

```
  //OBS: para alterar horário recebido por blynk no excel, basta usar essa fórmula.

  //=(ESQUERDA(xn;10)/60/60)/24+DATA(1970;1;1)+(tz/24) //onde xn é a célula com o valor recebido (exemplo: A1) e tz é a timezone (Fortaleza = -3)

  // --- APLICATIVO VX (onde X é um número a partir de 0) --- OBS: Aciona variável virtual VX

  BLYNK_WRITE(VX)

  { int       valorLido = param.asInt();

    //String  valorLido = param.asStr();

    //double  valorLido = param.asDouble();

  if (valorLido == 1)

  {

    param_calibrar = 1; //sabe que a calibração está ativada

    //CalibraSensor();

  }

  }//FIM BLYNK_WRITE(V5){

  // COMANDOS TERMINAL VIRTUAL


  // if you type "Marco" into Terminal Widget - it will respond: "Polo:"

  if (String("Marco") == param.asStr()) {

    terminal.println("You said: 'Marco'") ;

    terminal.println("I said: 'Polo'") ;

  } else {


    // Send it back

    terminal.print("You said:");

    terminal.write(param.getBuffer(), param.getLength()); //DEVOLVE IGUAL RECEBEU

    terminal.println();
```

```
}
// COMANDOS WIFI MANAGER
//declaração do objeto wifiManager
//WiFiManager wifiManager;

//utilizando esse comando, as configurações são apagadas da memória
//caso tiver salvo alguma rede para conectar automaticamente, ela é apagada.
//wifiManager.resetSettings();

//por padrão as mensagens de Debug vão aparecer no monitor serial, caso queira desabilitá-la
//utilize o comando setDebugOutput(false);
//wifiManager.setDebugOutput(false);

//caso queira iniciar o Portal para se conectar a uma rede toda vez, sem tentar conectar
//a uma rede salva anteriormente, use o startConfigPortal em vez do autoConnect
// wifiManager.startConfigPortal(char const *apName, char const *apPassword = NULL);
//entra no portal toda vez, ao inves de tentar se conectar a uma rede anterior.

//setar IP fixo para o ESP (deve-se setar antes do autoConnect)//Endereço AP
// setAPStaticIPConfig(ip, gateway, subnet);
// wifiManager.setAPStaticIPConfig(IPAddress(192,168,16,2), IPAddress(192,168,16,1),
IPAddress(255,255,255,0)); //modo AP
//wifiManager.setAPStaticIPConfig(IPAddress(10, 0, 0, 1), IPAddress(10, 0, 0, 1),
IPAddress(255, 255, 255, 0)); //se não configurar isso, cria uma rede 192.168.4.X

//setar IP fixo para o ESP (deve-se setar antes do autoConnect)//Endereço sendo CLIENTE
// setSTAStaticIPConfig(ip, gateway, subnet);
// wifiManager.setSTAStaticIPConfig(IPAddress(192,168,0,99), IPAddress(192,168,0,1),
IPAddress(255,255,255,0)); //modo estação
```

```
  //callback para quando entra em modo de configuração AP
  wifiManager.setAPCallback(configModeCallback);
  //callback para quando se conecta em uma rede, ou seja, quando passa a trabalhar em modo estação
  wifiManager.setSaveConfigCallback(saveConfigCallback);

  //=================================================================
  //---Cria uma rede WiFi
  //wifiManager.autoConnect("SSID", "senha"); //cria uma rede "SSID", com senha
  //wifiManager.autoConnect("WiFi_Manager"); //cria uma rede sem senha
  //wifiManager.autoConnect(); //gera automaticamente o SSID com o chip ID do ESP e sem senha

  //    wifiManager.setMinimumSignalQuality(10);  // % minima para ele mostrar no SCAN
  //    wifiManager.setRemoveDuplicateAPs(true);  //remover redes duplicadas (SSID iguais)
  //    wifiManager.setConfigPortalTimeout(20);   //timeout para o ESP nao ficar esperando para ser configurado para sempre
  //    wifiManager.setConnectTimeout(30);        //tempo para conexão
*/
```