UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PEDRO BRAGA LUZ

AUTOMATIZAÇÃO DE ESTUFA PARA CONTROLE DE AMBIENTE DE CULTIVO DE PLANTAS EM CONTEXTO RESIDENCIAL

FORTALEZA - CE
2022

PEDRO BRAGA LUZ

AUTOMATIZAÇÃO DE ESTUFA PARA CONTROLE DE AMBIENTE DE CULTIVO DE PLANTAS EM CONTEXTO RESIDENCIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Raphael Amaral da Câmara

FORTALEZA - CE

2022

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)

L994a Luz, Pedro Braga.
Automatização de estufa para controle de ambiente de cultivo de plantas em contexto residencial / Pedro Braga Luz. – 2022.
80 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2022.
Orientação: Prof. Dr. Raphael Amaral da Câmara.

1. Automatização. 2. Plantas. 3. Controle. 4. Internet das coisas. I. Título.

CDD 621.3

PEDRO BRAGA LUZ

AUTOMATIZAÇÃO DE ESTUFA PARA CONTROLE DE AMBIENTE DE CULTIVO DE PLANTAS EM CONTEXTO RESIDENCIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em: 05 de Julho de 2022

BANCA EXAMINADORA

Prof. Dr. Raphael Amaral da Câmara (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fabrício Gonzalez Nogueira
Universidade Federal do Ceará (UFC)

Eng. Gabriel Freitas Machado
Universidade Federal do Ceará (UFC)

Aos meus pais, por todo o suporte.

AGRADECIMENTOS

Aos meus pais, que me deram o apoio emocional e a educação para atingir meus objetivos de vida.

Aos meus irmãos, Moreno, Gabriel e Tiago, que são e sempre serão meus melhores amigos.

Aos meus amigos da UFC, que foram e são meus companheiros durante essa longa jornada, com quem compartilhamos os momentos de felicidade. Em especial ao grande amigo Gabriel, que foi de ajuda constante de solução de dúvidas e apoio técnico e moral na elaboração deste trabalho.

Ao meu Orietandor Prof. Dr. Raphael Amaral, por aceitar me orientar. Aos membros da banca pela disponibilidade e atenção.

A todos que contribuíram com a minha formação acadêmica e pessoal.

Os piores escravos são aqueles que estão servindo constantemente as suas paixões.

(Diógenes de Sínope)

RESUMO

Este trabalho consiste na montagem de um protótipo de estufa para controle de ambiente de cultivo de plantas em contexto residencial, com controle de variáveis de ambiente que influenciam no desenvolvimento das plantas. O trabalho se baseia no conceito de *Internet of Things* (IoT) para o acompanhamento e controle regulável de um ambiente de cultivo de plantas incluindo controle de fotoperíodo, umidade do solo (regagem), ventilação e temperatura ambiente. O projeto conta com um microcontrolador ESP-32 com programa responsável pelo controle dessas variáveis, bem como sensores periféricos para a aferição das variáveis de controle. Esse microcontrolador se comunica com a plataforma *IoT ThingSpeak* que acumula os dados do projeto para acompanhamento. Com esse tipo de controle, é possível que espécies de plantas não adaptadas a ambientes internos possam se desenvolver sem depender do ambiente externo.

Ao final do trabalho, são apresentados os resultados do projeto a partir das informações de funcionamento do protótipo, e será avaliado se o projeto desenvolvido atende as condições adequadas para o fim estabelecido, que é o controle automatizado de variáveis de cultivo em âmbito residencial.

Palavras-chave: automatização, plantas, controle, *Internet of Things*

ABSTRACT

This project consists in the assembly of a prototype greenhouse for controlling the cultivating environment for plants in a domestic situation, with controls for environment variables that influence plant growth and development. The project bases itself on the concept of Internet of Things (IoT) for the overseeing and regulation of the cultivation environment for plants, including controls for photoperiodism, soil humidity, ventilation and temperature. The project includes an ESP-32 microcontroller, which is programmed to control these variables and read their feedback loop through the use of peripheral sensors. This microcontroller comunicates with an IoT platform called ThingSpeak which stores the project's data. With this type of control, it's possible that plant species which are not adapted to indoors environments may develop without depending on the external environment.

At the end of this report, the results are presented based on the information gathered from the prototype in action, and they'll be evaluated to see if the developed project satisfies the conditions required for the end-goal, which is the automated control of plant cultivation variables in a domestic environment.

Keywords: automation, plants, control, *Internet of Things*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

A.C.	Antes de Cristo
TCC	Trabalho de Final de Curso
AC	Corrente Alternada
DC	Corrente Contínua
CO^2	Dióxido de Carbono
O^2	Oxigênio
ž*C*	Graus Celsius
nm	Nanômetros
MHz	Megahertz
MB	Megabytes
I/O	*Input/Output*
UART	*Universal asynchronous receiver/transmitter*
ADC	Conversor analógico-digital
V	Volts
l/h	Litros por hora
W	Watts
A	Ampére
VDC	Volts de corrente contínua
RTC	*Real-time clock*
RTOS	*Real-time Operating System*
IoT	*Internet of Things*
API	Interface de programação de aplicações
GND	Terra (*Ground)*
LED	Diodo emissor de luz

SUMÁRIO

1 INTRODUÇÃO

Durante a maior parte da existência do homem moderno, ou Homo Sapiens, tinham como estilo de vida a caça e a coleta em busca de alimentação. (HISTORY, 2018) O cultivo de plantas realizado por humanos data de mais de 12.000 anos atrás, por volta do ano 10.000 A.C., quando ocorreu a "Revolução Neolítica", também conhecida como revolução agrícola. Essa é considerada uma das mudanças mais importantes no estilo de vida da civilização humana, modificando como a nossa espécie se alimentava, interagia com o meio ambiente e socializava entre si. (BLAKEMORE, 2019)

Desde então, a agricultura se tornou uma das atividades mais básicas e essenciais à manutenção da vida humana no nosso planeta. Hoje, o cultivo de plantas é praticado por diversos motivos, não apenas para a produção de alimentos, mas também como forma de tornar ambientes mais bonitos e agradáveis, além de também ser bastante praticada como um passatempo. Durante a pandemia, por exemplo, a jardinagem se tornou uma forma de terapia para muitas pessoas que passavam por problemas psicológicos, aumentando em até 50% as vendas de viveiros de plantas. (ROPELLI, 2020)

É notável que o cultivo de plantas requer diversos cuidados específicos e manutenção constante. Toda planta possui um habitat natural e condições ideais distintas para o seu desenvolvimento e crescimento, como períodos de iluminação durante o dia (ou fotoperíodos), temperaturas, níveis de umidade e irrigação, onde cada espécie de planta possui necessidades específicas de cada uma dessas variáveis, onde a planta pode se desenvolver de melhor forma quando essas variáveis são otimizadas.

O morango, por exemplo, é uma fruta que necessita de fotoperíodos e temperaturas distintos em cada fase do seu desenvolvimento. Durante a fase de formação de mudas, por exemplo, a temperatura média diurna deve ser entre 20 ºC e 26 ºC e menor que 15 ºC durante a noite, sendo essa temperatura ótima variável durante o processo de desenvolvimento do morangueiro.(ANTUNES et al., 2016) Esse tipo de condição específica faz com que o morango e diversas outras frutas ou vegetais só sejam produzidos em regiões específicas e durante épocas específicas de um ano, como podemos ver na figura abaixo com a produção do morango no Brasil sendo concentrada nas regiões Sul e Sudeste do país.

Tabela 1 – Área, produção e produtividade de morango no Brasil em 2010.

Estado	Área (ha)	Produção (t)	Produtividade (t ha^{-1})	Participação (%)
Minas Gerais	1.790	72.716	41	54,52
Paraná	600	18.000	30	13,50
Rio Grande do Sul	500	15.000	30	11,25
São Paulo	331	10.655	32	7,98
Espírito Santo	240	8.000	33	5,99
Distrito Federal	120	4.800	40	3,59
Rio de Janeiro	37	2.220	60	1,67
Santa Catarina	100	2.000	20	1,50
Total	**3.718**	**133.391**	-	**100,00**

Fonte: EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA (EMBRAPA).

A manipulação dessas variáveis com o intuito de praticar a agricultura em ambientes não ortodoxos é praticada pela humanidade a diversos séculos. Existem registros em tumbas egípcias que indicam que os faraós tentaram cultivar frutas em jardins em ambientes fechados. Com o passar dos anos, novas ferramentas tecnologias foram sendo desenvolvidas e se tornando acessíveis, avançando cada vez mais as técnicas para a agricultura e horticultura em ambientes cada vez mais abrangentes. O vidro, por exemplo, começou a ser utilizado por volta do século XVI para a construção de estufas que serviam como ambientes fechados mais controláveis do que o cultivo em ambientes abertos. (HARDIGREE, 1980)

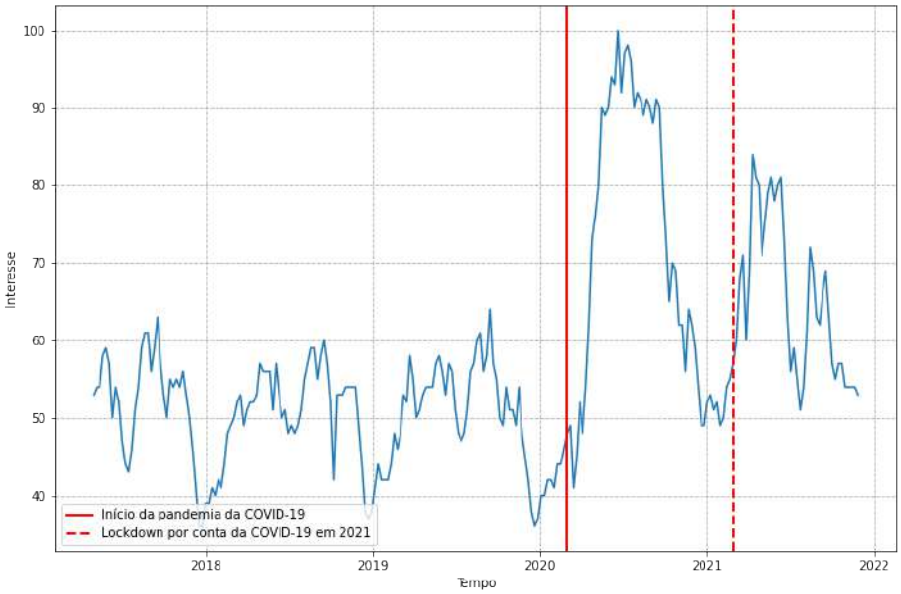
Hoje, com o avanço da tecnologia, é possível controlar cada vez mais o ambiente de cultivo e tornar o processo mais automatizado para haver menor necessidade de manutenção. O cultivo que antes era restrito a regiões, climas e épocas do ano específicas agora pode ser viável em praticamente qualquer ambiente, dado o uso das ferramentas e técnicas corretas. Desde o cultivador de apartamento com pouco acesso a luz solar querendo cultivar frutas e vegetais em pequena escala até grandes produções de frutas em estufas modernas e sofisticadas podem utilizar técnicas para realizar o cultivo das mais diversas plantas nos mais diversos ambientes.

1.1 Objetivo principal do estudo

Hoje, o cultivo de plantas no âmbito doméstico tem se tornado um interesse cada vez maior da população brasileira, ganhando grande popularidade nos últimos anos

e em especial durante a pandemia da COVID-19, conforme observamos na figura abaixo.

Figura 1 – Interesse por plantas ao longo dos últimos anos durante a pandemia de COVID-19 no Brasil. (de 0 a 100, onde 100 representa o pico de interesse)



Fonte: (GOOGLE, 2022), adaptado pelo autor.

Este TCC tem por objetivo principal a apresentação e montagem de uma estufa como ambiente controlado e automatizado para o cultivo de plantas em um contexto residencial para pequena escala. O intuito é utilizar técnicas de controle, automação e IoT para criar um ambiente adequado não só para plantas ornamentais, que, na maioria das vezes, são consideravelmente mais fáceis de se cultivar (HARDIGREE, 1980), mas de fornecer o ambiente necessário para a monitoração e cultivo de plantas que normalmente não seriam adaptadas para ambientes domésticos ou internos.

1.2 Estrutura

A estrutura a ser utilizada neste trabalho apresentará a seguinte forma:

- O Capítulo 1 é uma apresentação geral do estudo buscando trazer a familiarização com o tema, os objetivos deste trabalho e a estrutura deste estudo;

- O Capítulo 2 estabelece a fundamentação teórica necessária para o desenvolvimento do projeto. Desde as características importantes da cultivação de plantas até o embasamento tecnológico necessário para a criação do protótipo;
- O Capítulo 3 é a explicação da metodologia para desenvolvimento do protótipo, mostrando os componentes utilizados no projeto e as razões pelas quais esses foram escolhidos, juntamente com a apresentação do passo-a-passo realizado na montagem e com a explicação dos códigos e da lógica de programação utilizada;
- O Capítulo 4 apresenta os resultados obtidos com a implementação apresentada no capítulo anterior
- O capítulo 5 é a reflexão final sobre o projeto, refletindo sobre os resultados, apresentando justificativas para o funcionamento do que foi proposto, sugestões de possíveis mudanças ou alterações e, finalmente, avaliando se o resultado foi satisfatório de acordo com a proposta do projeto

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo será estabelecida a fundamentação teórica necessária para desenvolver este TCC. Serão apresentados os aspectos gerais das plantas, como introdução para técnicas de cultivo de plantas em ambientes controlados, conforme a literatura. É válido notar que a estrutura e o ciclo de vida das plantas é extremamente complexo e, por isso, apenas algumas características gerais e importantes para o cultivo serão abordadas nessa fundamentação teórica.

2.1 Características gerais das plantas

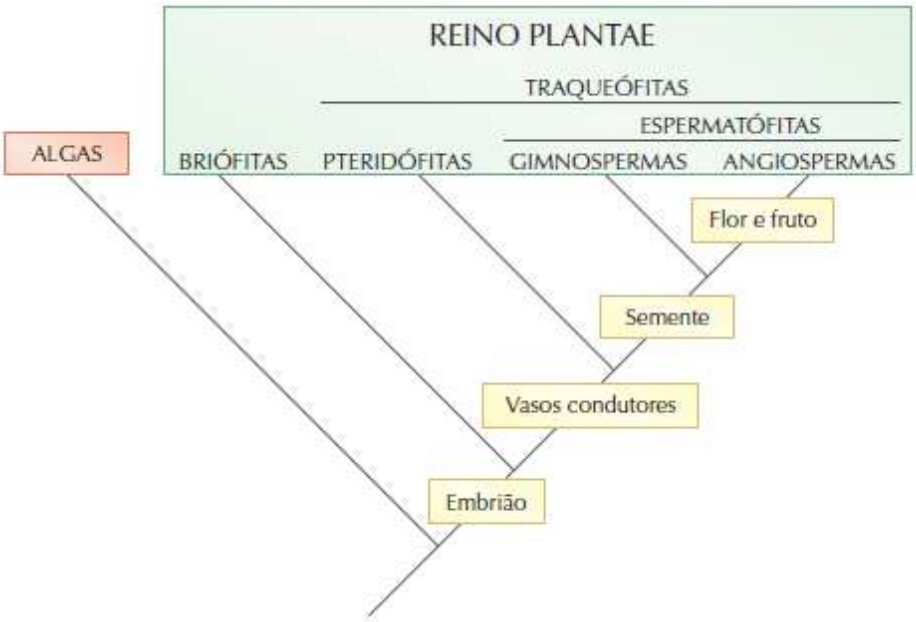
As plantas são organismos pertencentes ao reino Plantae, cuja principal característica é a capacidade de realizar o processo de fotossíntese, ou seja, a capacidade de realizar processos químicos utilizando a luz para a geração de substâncias orgânicas que lhes servem como alimento (AMABIS, MARTHO, 2009). O reino Plantae é dividido em diversos grupos com características diferentes que são compartilhadas pelos grupos mais recentes na escala evolutiva, sendo os principais grupos vistos na figura **??** e descritos a seguir:

- Briófitas: plantas pequenas e delicadas que vivem geralmente em ambientes úmidos e sombreados, como barrancos e troncos de árvores no interior das matas. Tem como principal característica a ausência de vasos condutores de seiva. Representada principalmente por musgos. (AMABIS, MARTHO, 2009)
- Pteridófitas: Caracterizadas principalmente por não formar sementes e por apresentarem estruturas de vasos como o xilema, responsável por transportar água e sais minerais das raízes até as folhas, e o floema, responsável por transportar uma solução de açúcares e compostos orgânicos produzidos nas folhas para as demais partes da planta. Representada principalmente pelas samambaias. (AMABIS, MARTHO, 2009)
- Gimnospermas: A principal novidade em relação às pteridófitas é a presença da semente. A maior parte das espécies desse grupo são árvores grandes denominadas coníferas, caracterizadas por terem estruturas reprodutivas em formato de cone. Representada principalmente pelos pinheiros ou

araucárias. (AMABIS, MARTHO, 2009)

- Angiospermas: São o grupo dominante de plantas no planeta e constituem a maior parte da vegetação, além da maior variedade de espécies entre os grupos. Possuem como principal novidade em relação às gimnospermas a presença de flores e frutos, sendo o único grupo com a presença dessas estruturas. Representadas por diversos tipos de plantas diferentes, desde espécies com menos de 1 milímetro de comprimento até árvores com mais de 100m de altura, e ocupando os mais diversos ambientes, como o solo, a água ou até sendo parasitas ou "inquilinas". (AMABIS, MARTHO, 2009)

Figura 2 – Cladograma com características compartilhadas pelos grupos de plantas



Fonte: (AMABIS, MARTHO, 2009)

As angiospermas são o grupo de maior importância econômica e social para a humanidade já que a maior parte das plantas arbóreas ou hortícolas produtoras de frutos para fins de alimentação, bem como as plantas que criam substâncias extraídas para fins medicinais, fazem parte deste grupo, principalmente pela grande diversidade de espécies encontradas nas angiospermas. Assim, o foco de informações teóricas apresentadas daqui para a frente será com relação ao grupo das angiospermas.

2.1.1 Fotossíntese

A fotossíntese é o processo realizado pelas plantas capaz de utilizar a energia da luz solar, juntamente com o gás carbônico proveniente do ar e a água proveniente do solo para obtenção da energia metabólica. Com esse processo, são formados oxigênio e glicídios, principalmente a sacarose e o amido, que serão futuramente convertidos nas diversas substâncias necessárias para os processos químicos das plantas.(AMABIS, MARTHO, 2009).

Juntamente à fotossíntese, as plantas realizam um processo chamado de respiração celular para a manutenção de suas células. Esse processo utiliza dos sacarídios gerados pelas plantas e do oxigênio para a produção de energia, liberando no processo gás carbônico e água, que são utilizados novamente para o processo de fotossíntese. As trocas gasosas envolvidas nesses dois processos são realizadas por estruturas denominadas estômatos, que são como poros nas folhas das plantas que se abrem ou fecham de acordo com fatores ambientais, sendo estes fatores: Luminosidade, concentração de gás carbônico e suprimento hídrico.

Quadro 1 – Fatores ambientais envolvidos na abertura e fechamento dos estômatos

Condições ambientais		Comportamento do estômato
Intensidade luminosa	Alta	Abre
	Baixa	Fecha
Concentração de CO_2	Alta	Fecha
	Baixa	Abre
Suprimento de água	Alto	Abre
	Baixo	Fecha

Fonte: (AMABIS, MARTHO, 2009)

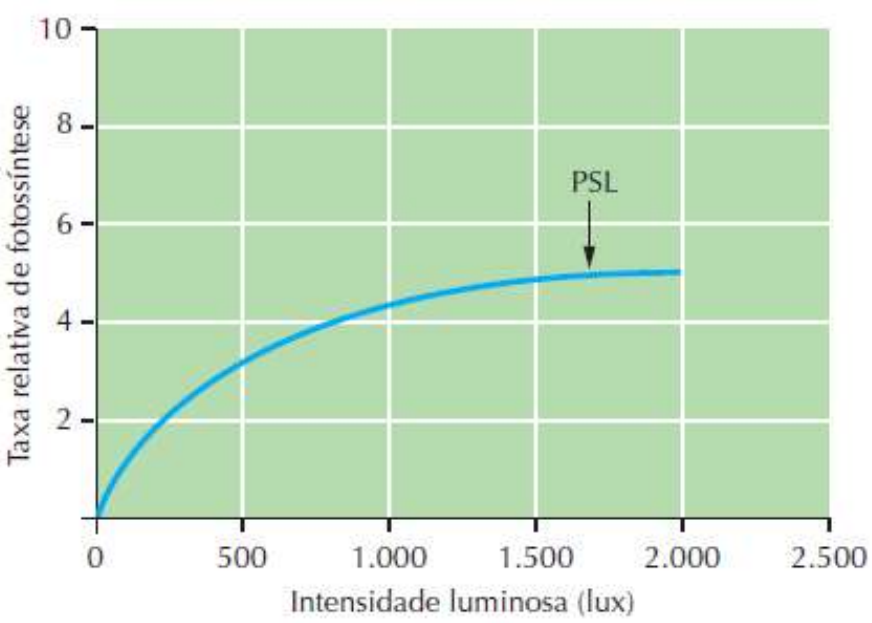
Dentre os fatores que afetam o processo de fotossíntese em si, os principais a ser considerados são:

- Concentração de CO^2 : $A concentração de gás carbônico no ar atmosférico influencia diretamente$ atmosférica aumenta, até que esta atinja cerca de 10 vezes a concentração atmosférica normal, não havendo aumento nessa taxa a partir daí.
- Temperatura: Cada espécie de planta possui temperaturas adequadas para sua sobrevivência. A taxa fotossíntese no entanto está diretamente

relacionada ao aumento da temperatura, até uma temperatura ambiente de cerca de 35 ºC, sendo este um limite que, se superado, pode causar drástica redução da fotossíntese e afetar negativamente diversos outros processos biológicos das plantas.

- Luminosidade: A taxa de fotossíntese aumenta proporcionalmente ao aumento de luminosidade, até certo valor-limite, denominado ponto de saturação luminosa

Figura 3 – Efeito do aumento da intensidade luminosa sobre a fotossíntese

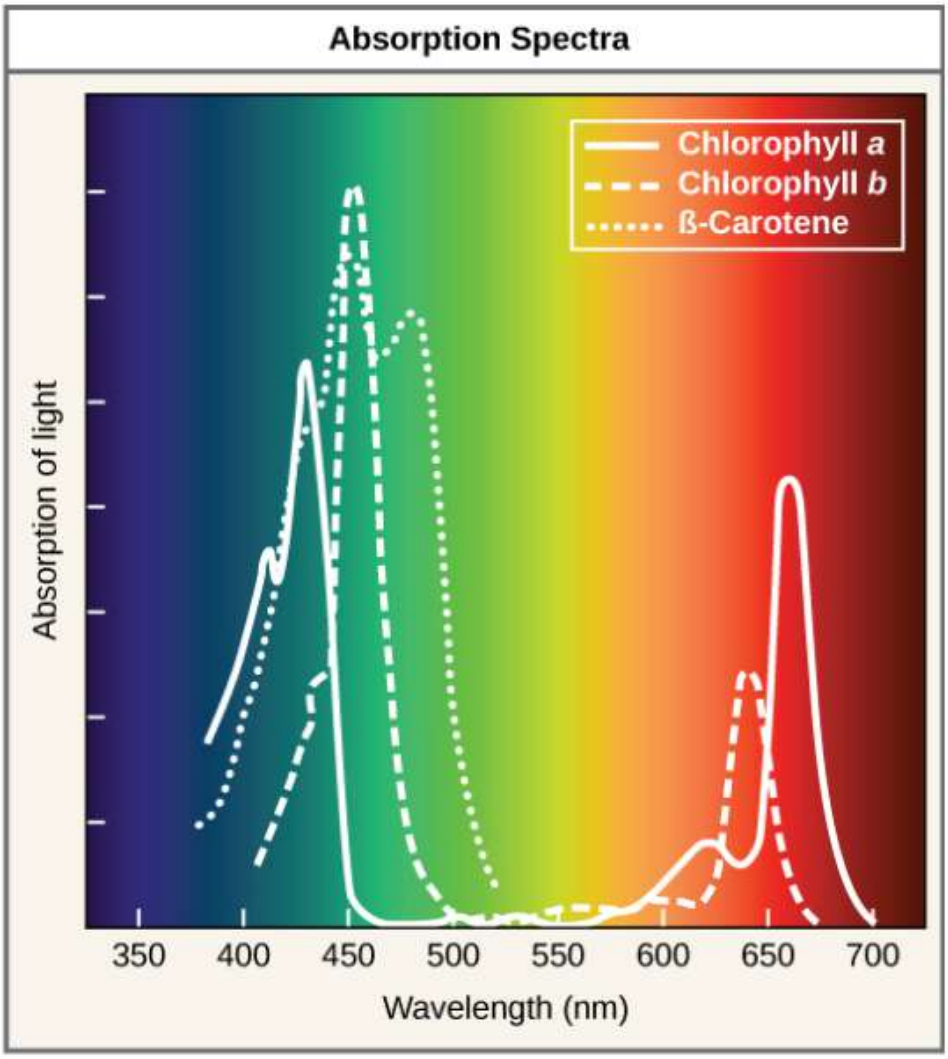


Fonte: (AMABIS, MARTHO, 2009)

Ainda em relação a luminosidade, é importante tratar sobre o tipo de luz absorvido por plantas para o processo de fotossíntese. O sol emite radiação eletromagnética em diversos comprimentos de onda, onde cada um desses comprimentos de onda possuem energias características. Toda a radiação eletromagnética, incluindo a luz visível, é caracterizada pelo seu comprimento de onda. Esses comprimentos de onda são absorvidos pelos organismos por pigmentos, que são substâncias que reagem à luz dependendo do nível de energia que esses pigmentos são capazes de absorver. (RYE, 2016)

No caso das plantas, os principais pigmentos fotossintéticos encontrados nesses organismos são da classe de clorofilas e carotenóides (RYE, 2016). Alguns exemplos de espectros de absorção de pigmentos dessas classes podem ser observados na figura abaixo.

Figura 4 – Espectro de absorção de luz para: Clorofila A, Clorofila B e Beta-caroteno



Fonte: (RYE, 2016)

Em geral, os melhores comprimentos de onda para o processo de fotossíntese ficam em torno da luzes visíveis Azul (425nm a 450nm) e Vermelha (600nm a 700nm) (VERNIER, 2018).

2.1.2 Nutrição mineral das plantas

O gás carbônico, a água e os sais minerais são todos os elementos químicos necessários para o funcionamento das celúlas das plantas. A água e os sais minerais são absorvidos do solo pelas raízes das plantas e constituem a denominada nutrição mineral das plantas. Dentre os elementos químicos absorvidos, alguns são necessários em grandes quantidades, denominados macroelementos, enquanto outros são necessários em

quantidades pequenas, sendo por isso denominados de microelementos.

Quadro 2 – Elementos químicos essenciais às plantas

Macroelementos		Microelementos	
Hidrogênio	(H)	Cloro	(Cℓ)
Carbono	(C)	Ferro	(Fe)
Oxigênio	(O)	Boro	(B)
Nitrogênio	(N)	Manganês	(Mn)
Fósforo	(P)	Sódio	(Na)
Cálcio	(Ca)	Zinco	(Zn)
Magnésio	(Mg)	Cobre	(Cu)
Potássio	(K)	Níquel	(Ni)
Enxofre	(S)	Molibdênio	(Mb)
Silício	(Si)		

Fonte: (AMABIS, MARTHO, 2009)

A deficiência de algum elemento químico essencial pode apresentar sintomas específicos da deficiência. A falta de magnésio por exemplo deixa as folhas amareladas e diminui a produção de clorofila, atrapalhando o processo da fotossíntese. As deficiências nutricionais mais comuns são as dos elementos nitrogênio, fósforo e potássio AMABIS, MARTHO (2009). Podemos ver alguns exemplos de sintomas observáveis devido a falta de nutrientes na figura abaixo.

Figura 5 – Exemplos de sintomas de faltas de nutrientes

Fonte: (MOREIRA, 2018)

A absorção de água e nutrientes pelo solo é realizada pelas raízes através de estruturas denominadas folículos, que possuem grande área de superfície e realizam essa absorção por osmose. O produto de água e nutrientes absorvido pela planta recebe o nome de seiva bruta e é transportada pelo xilema para o resto das estruturas da planta(RUEHR, 2018).

Existe também um processo realizado pelos estomas, estruturas mencionadas anteriormente, que ficam nas folhas e realizam a respiração celular. Essas estruturas também realizam a transpiração, regulando a umidade e temperatura da planta e contribuindo para a geração de pressão osmótica nas raízes e ajudando na absorção de água e nutrientes (RUEHR, 2018)

2.2 Controle

Ao longo da humanidade, o avanço da tecnologia foi tornando processos que antes eram difíceis cada vez mais eficientes e independentes da ação humana. O controle

automatizado é o nome dado para o conjunto de ferramentas que são capazes de regular um processo para torná-lo mais eficiente e simples. Sistemas de controle são uma ferramenta integral na sociedade moderna, e existem amplos exemplos de aplicação da aplicação de técnicas de controle automatizado em diversas áreas da ciência e da indústria (NISE, 2013).

Essencialmente, um sistema de controle é um conjunto de subsistemas e processos unidos com o propósito de controlar uma variável e gerar uma saída desejada com base em uma entrada desejada, com as características de performance desejada. A figura abaixo mostra uma simplificação de um sistema de controle.

Figura 6 – Descrição simplificada de um sistema de controle

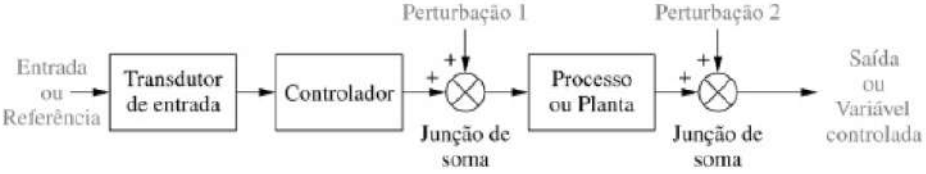
Fonte: (NISE, 2013)

Quando falamos de sistemas de controle, podemos dividi-los em duas grandes configurações: Sistemas de malha aberta e Sistemas de malha fechada. Para esse projeto, o controle de malha aberta descreve melhor o funcionamento do sistema do protótipo elaborado.

2.2.1 Sistemas de malha aberta

Sistemas de controle de malha aberta são os sistemas que não possuem retroalimentação do sinal de saída. Sendo assim, sistemas de malha aberta não são capazes de modificar sua forma de atuação de acordo com a saída obtida.

Figura 7 – Sistema de controle de malha aberta



Fonte: (NISE, 2013)

Conforme mostra a figura 7, a principal característica de um sistema de malha

aberta é que ele não é capaz de compensar por distúrbios ou perturbações adicionadas ao sinal do controlador, e são puramente comandados pelo sinal de entrada (NISE, 2013).

2.3 Cultivo em estufas controladas

É importante definir os fatores importantes para o cultivo de plantas com base no que já foi apresentado, e mostrar as técnicas e instrumentos de controle utilizados por diferentes iniciativas de ambientes de cultivo automatizados.

2.3.1 Iluminação

A iluminação é o fator mais essencial para o funcionamento do organismo das plantas e para sua sobrevivência. As estufas tradicionais utilizam vidros. Para o cultivo em ambientes fechados sem acesso à luz solar, é necessário que haja a utilização de iluminação artificial.

Antigamente, as lâmpadas incadescentes eram a única forma de gerar a iluminação para o cultivo (HARDIGREE, 1980). Hoje, as lâmpadas fluorescentes e LEDs são as mais utilizadas para esse tipo de aplicação. As lâmpadas fluorescentes possuem a vantagem do custo baixo de aquisição e são mais eficientes que as lâmpadas incandescentes. No entanto, os LEDs, apesar de possuirem custo inicial alto, são a opção mais eficiente energeticamente, além de possuirem a grande vantagem de serem "adaptáveis"em relação aos comprimentos de onda produzidos por eles, o que pode tornar o cultivo ainda mais otimizado (BARNITZ, 2022).

O controle do fotoperíodo é um dos aspectos mais importantes com relação ao controle do sistema de iluminação de um cultivo. Cada espécie de planta pode necessitar de diferentes períodos de iluminação durante o dia, e isso também pode variar para cada fase do desenvolvimento da planta a ser cultivada (HARDIGREE, 1980).

2.3.2 Umidade

A água é um dos fatores limitantes da produção agrícola, considerando sua participação nos vários processos metabólicos da planta. O excesso de água, no entanto, pode afogar as raízes, o que limita a respiração e por sua vez a fotossíntese, podendo levar a problemas ou até a morte da planta. É necessário portanto, que haja o controle

cuidadoso do nível de umidade do solo.

Em geral, são utilizadas bombas de água como atuadores para o controle da irrigação, sendo a irrigação realizada em várias técnicas diferentes, como a nebulização, o gotejamento ou a aspersão.

2.3.3 Ventilação

A concentração dos gases O^2 e CO^2 no ar é importante para a realização das trocas gasosas nos processos de fotossíntese e respiração celular. Em ambientes fechados, é necessário que haja a troca do ar com o ambiente externo para a troca desses gases, assim como para a manutenção da temperatura e evitar a proliferação de pestes.

A utilização de ventiladores é recomendável para estufas, tanto em pequena escala quanto em média ou grande escala. Existem sensores de níveis de concentração de gases que também são utilizados no controle de estufas agrícolas, geralmente para aplicações em grande escala, como o que pode ser visto na figura abaixo.

Figura 8 – Sensor de concentração de gás carbônico

Fonte: (NISE, 2013)

2.3.4 Temperatura

A manutenção de uma faixa de temperatura adequada para plantas é importante para que os processos metabólicos possam ocorrer da forma adequada. O controle

da temperatura é dependente do ambiente no qual o cultivo está situado podendo ou não ser necessário o controle dessa variável, mesmo para ambientes internos.

Para o acompanhamento da temperatura, existem inúmeros sensores de temperatura que são utilizados em todo tipo de aplicação, de pequena, média ou grande escala. Para o controle da temperatura em si, é necessário o uso de climatizadores que possam manter a temperatura do ambiente.

3 METODOLOGIA

Esse capítulo explicará o passo a passo do projeto do sistema de automatização proposto, começando por todos os utilitários físicos que foram utilizados como microcontrolador, sensores e atuadores, em seguida mostrando as ferramentas de software utilizadas.

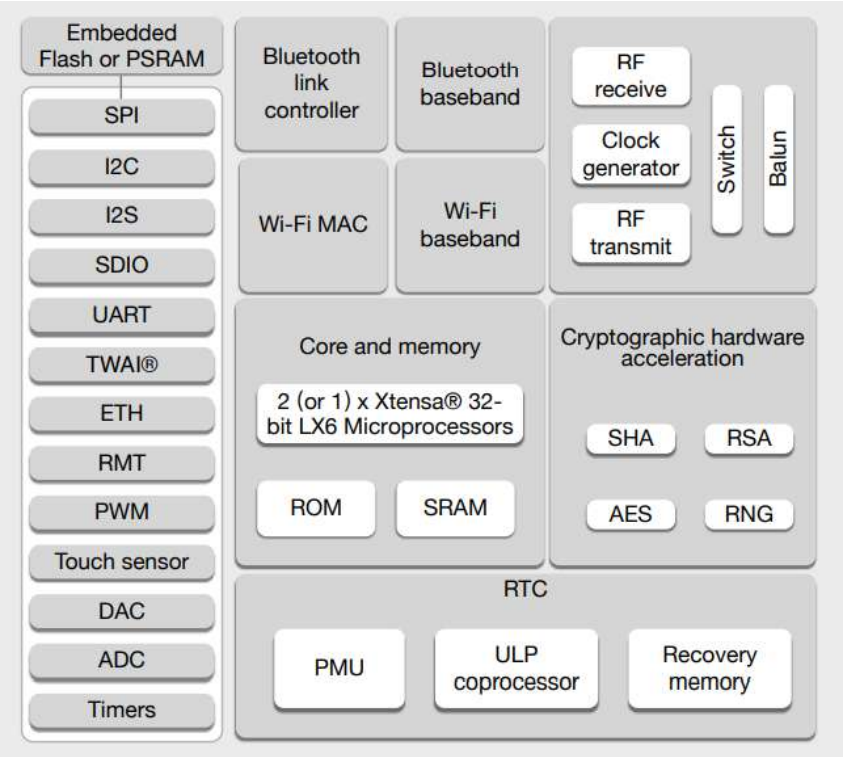
3.1 Hardware

Nessa seção, serão mostrados os componentes utilizados na montagem do projeto, assim como as razões pelas quais esses componentes foram escolhidos.

3.1.1 Microcontrolador: ESP32

Para este projeto, foi escolhido o microcontrolador ESP32 da ESPRESSIF Microsystems. Esse microcontrolador apresenta um processador Xtensa Dual-Core LX6 de 32 bits, com *Clock* de 160MHz até 240MHz. O modelo do módulo utilizado é o ESP-WROOM-32D, com memória *Flash* de 4MB. Além disso, o controlador possui capacidade de conexão *WiFi*, característica importante para o projeto. Podemos observar as características funcionais do ESP32 na figura abaixo.

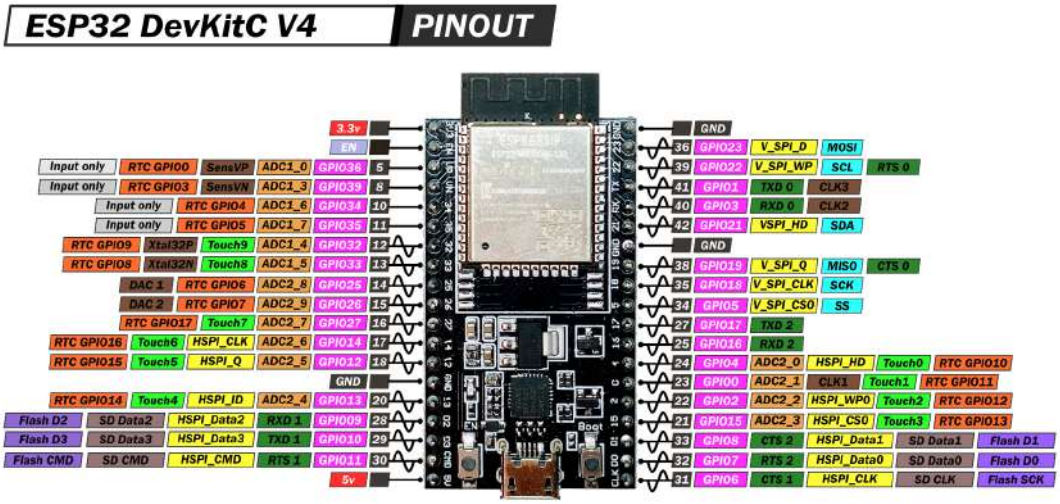
Figura 9 – Diagrama de blocos funcional do ESP32



Fonte: (ESPRESSIF)

Para esse projeto, foi utilizada a placa de desenvolvimento ESP32 DevKitC V4, fabricada pela própria ESPRESSIF. Essa placa possui os pinos de I/O saltados para fácil acesso, além de porta Micro-USB para alimentação e para acesso ao *UART* do microcontrolador para comunicação *Serial*. O modelo utilizado e seu diagrama de pinos podem ser vistos na figura abaixo.

Figura 10 – Diagrama de pinos da placa ESP32 DevKitC V4



Fonte: (MISCHIANTI, 2021)

3.1.2 Sensor de umidade do solo

O sensor para detecção de umidade do solo escolhido foi o sensor produzido pela ICStation (visto na figura 11), com capacidade de operação de 3.3V - 5V, ideal para o uso com o ESP32. Possui saída digital, regulada por um potenciômetro que pode ser calibrado para determinar um nível limite de umidade onde a saída digital é acionada, e possui também uma saída analógica, que gera um valor de tensão de 0 a tensão de alimentação. É ideal para o uso com o ESP32 que possui conversor ADC de 0-3.3V com resolução de 12 bits. Para o projeto, será utilizada apenas a saída analógica, para acompanhamento da umidade relativa do solo.

Figura 11 – Sensor de umidade do Solo

Fonte: (ICstation)

3.1.3 Sensor de Temperatura

Para a medição da temperatura do sistema, foi utilizado o sensor de temperatura e umidade SHT20. Esse sensor possui capacidade de medição da escala de -40°C até 125°C, com resolução de 0.04°C e precisão típica em torno de 0.03°C na faixa de operação de 0°C a 60°C, como mostrado na figura 13, faixa essa que inclui as temperaturas para a sobrevivência da maioria das plantas. O sensor e suas características podem ser vistos nas figuras 12 e 13.

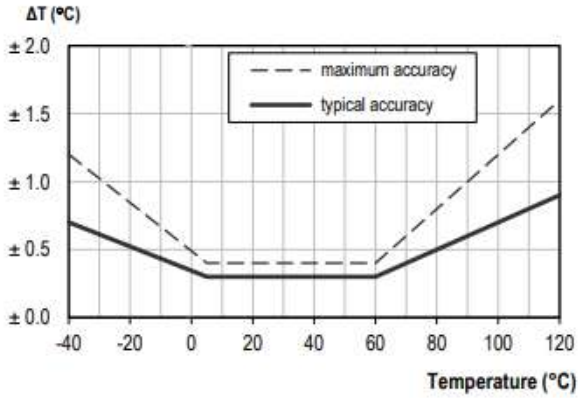
Figura 12 – Sensor de temperatura e umidade do ar SHT20

Fonte: (Sensirion, 2014)

Figura 13 – Especificações e gráfico de precisão típica do sensor SHT20

Temperature

Parameter	Condition	Value	Units
Resolution [1]	14 bit	0.01	°C
	12 bit	0.04	°C
Accuracy tolerance [2]	typ	±0.3	°C
	max	see Figure 3	°C
Repeatability		±0.1	°C
Operating Range	extended [4]	-40 t0 125	°C
Response Time [7]	τ 63%	5 to 30	s
Long Term Drift[8]	Typ.	< 0.02	°C/yr



Fonte: (Sensirion, 2014)

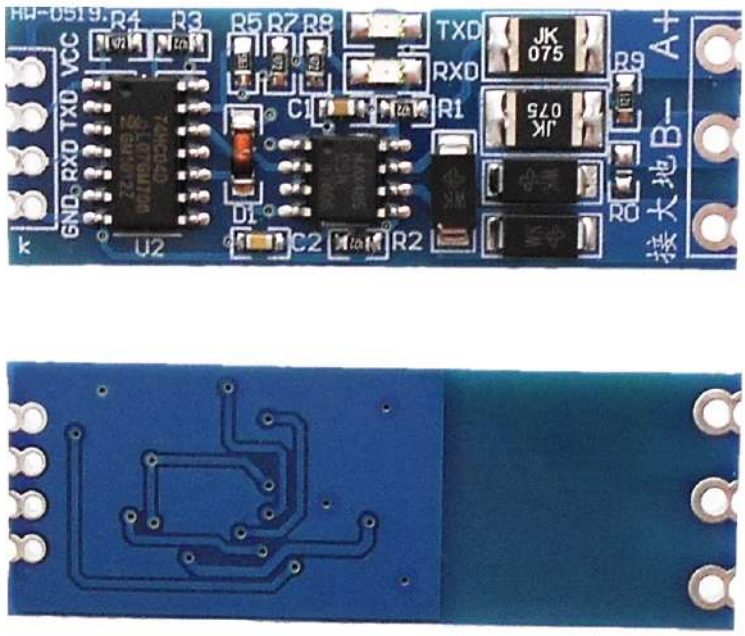
O sensor encontra-se integrado em um módulo RS485 para a utilização do protocolo *Modbus*, que gera comunicação em série confiável entre o sensor e o ESP32. O módulo pode ser observado na figura 14.

Figura 14 – Módulo SHT20 com módulo RS485

Fonte: (WorldChips)

Para a comunicação com o controlador, é necessário a conversão do sinal RS485 para TTL. Essa conversão foi feita utilizando o conversor RS485 para TTL MAX485, integrado no módulo HW 0519 mostrado na figura 15.

Figura 15 – Módulo HW 0519 para conversão RS485-TTL

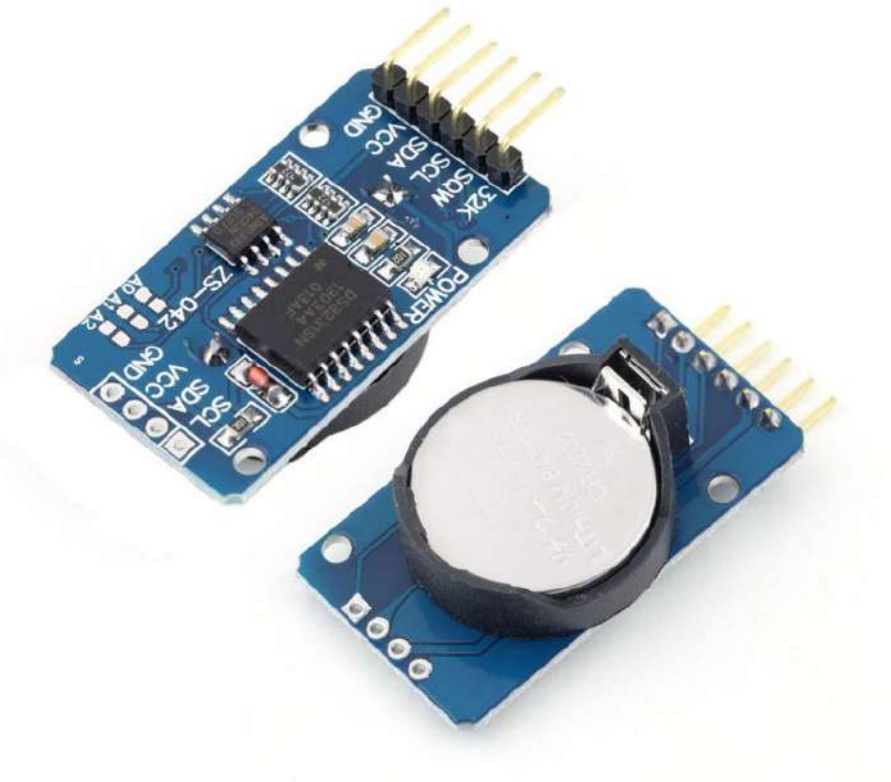


Fonte: (Jisda)

3.1.4 Módulo RTC

Para o controle da iluminação, será utilizado um módulo *Real-time-clock* como forma de sincronização do controlador ao horário atual do dia. Para o protótipo, foi utilizado o módulo DS3231 (figura 16 de baixo custo e extrema precisão da *Maxim Integrated*.

Figura 16 – Módulo RTC DS3231



Fonte: (Integrated, 2015)

Esse módulo pode ser alimentado com 3.3V e possui diversas funções para leitura do tempo, como leitura de dia do mês, dia da semana, hora do dia e diversas outras funções, além de bateria própria para manutenção da configuração em caso de falta de energia (Integrated, 2015).

3.1.5 Bomba de Água

Para o controle da umidade do solo, será utilizada uma bomba de água para a rega das plantas. A bomba de água escolhida foi uma bomba de aquário submersa do modelo EL-P310. A figura abaixo mostra a bomba utilizada.

Figura 17 – Bomba de aquário submersa

Fonte: (ELO Imports)

Possui vazão máxima de até 300l/h e altura máxima da coluna de água de 60cm conforme mostra a figura 18, especificações estas que serão adequadas para o projeto.

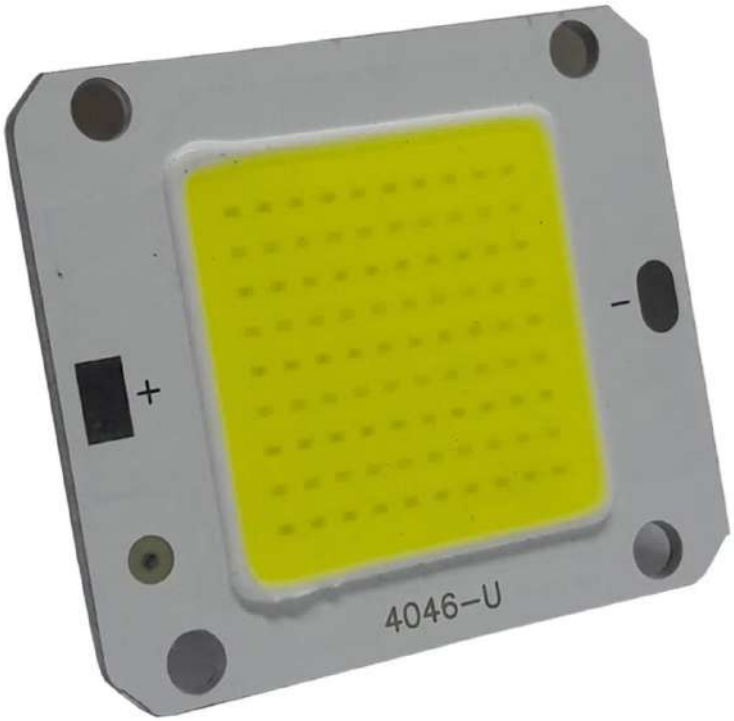
Figura 18 – Especificações da bomba EL-P310

Fonte: (ELO Imports)

3.1.6 LED

A iluminação utilizada no projeto é feita pela utilização dos chips *Chip-on-Board* 4046-U, de potência de até 30W, que podem ser vistos na figura abaixo.

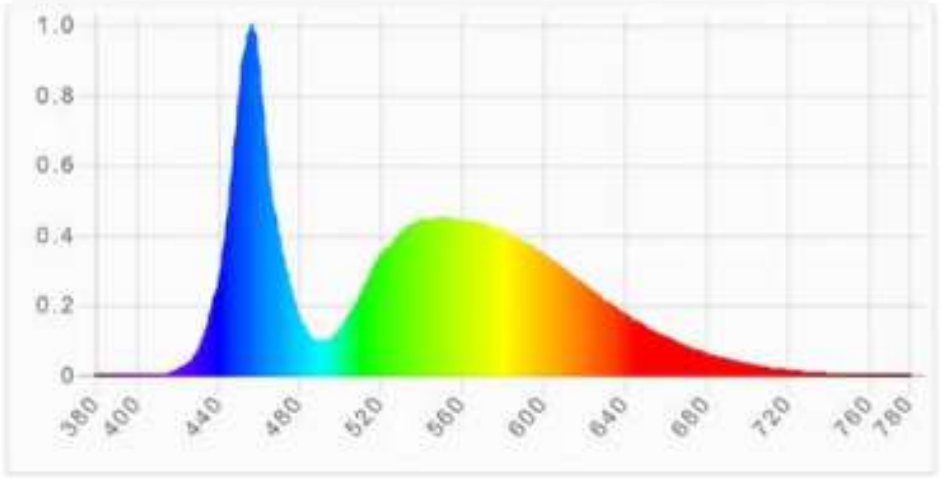
Figura 19 – Chip 4046-U



Fonte: (Carrefour)

Esses chips emitem luz branca de espectro suficientemente adequado para a utilização no cultivo de plantas apesar da baixa intensidade da luz vermelha emitida por esses, pois possuem grande concentração de luz azul, como observamos na figura 20.

Figura 20 – Espectro de luz para LED branco



Fonte: (CCL, 2018)

Os LEDs são alimentados por corrente contínua através de reatores que fazem a conversão da tensão da rede de 220V 60Hz para a tensão de 20V em corrente contínua. Os reatores podem ser vistos na figura abaixo.

Figura 21 – Reator dos LEDs

Fonte: Próprio autor

3.1.7 Ventiladores

Para o controle da temperatura ambiente e para a circulação de gás carbônico e oxigênio no ambiente de cultivo, serão utilizados ventiladores coolers sem escova, como os modelos utilizados em gabinetes de computador. São ventiladores com motor sem escova que operam sob tensão contínua de 12V e corrente máxima de até 0.2A. Uma ventoinha dessa forma pode ser vista na figura abaixo.

Figura 22 – Ventoinha 12V

Fonte: Próprio autor

3.1.8 Módulo de Relés

Para o controle dos atuadores mencionados nas seções anteriores, será utilizado um módulo de relés de 4 canais (figura 23) alimentado por 5VDC, com capacidade de controle de cargas de até 220V e correntes de até 10A.

Figura 23 – Módulo de Relés 4 canais

Fonte: (Teles Componentes)

As saídas dos relés serão controladas por pinos de saída do ESP32 através do software do protótipo.

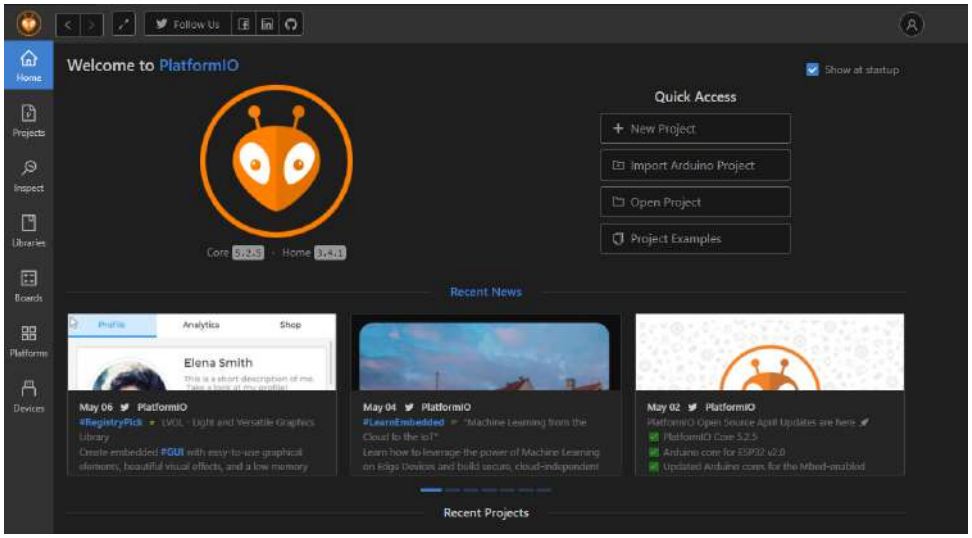
3.2 *Software*

A seguir será mostrado o *software* que foi utilizado para a parte programável do sistema proposto, bem como o software de acompanhamento *IoT* que armazena os dados do sistema em produção.

3.2.1 IDE: PlatformIO

A IDE utilizada para o desenvolvimento do código do projeto foi o *PlatformIO* (mostrada na figura abaixo), uma extensão do Microsoft Visual Studio Code com suporte para diversos frameworks e que tem o intuito de acelerar e simplificar a criação de código para sistemas embarcados.

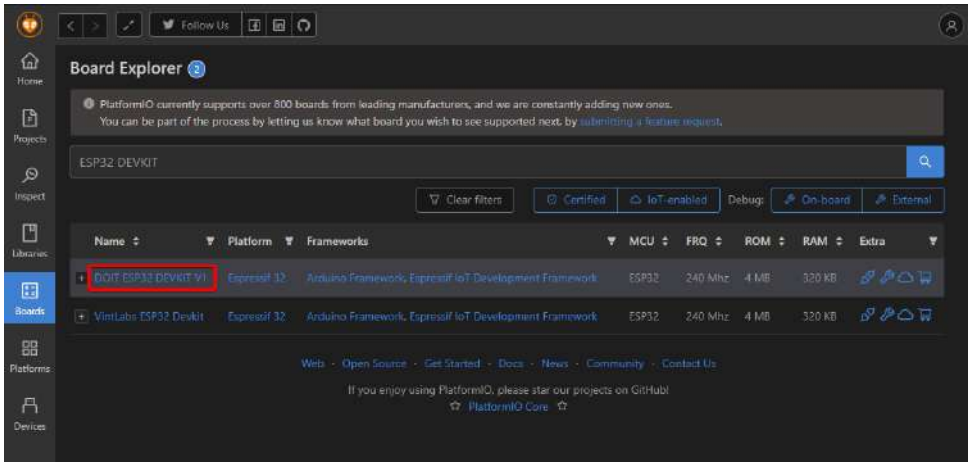
Figura 24 – Tela Inicial do *PlatformIO*



Fonte: Próprio autor.

Para a geração do projeto com as características apropriadas, é necessário inicialmente instalar as dependências associadas a placa que será utilizada no projeto. Para isso, foi acessada a aba *Boards*, onde lá é possível pesquisar por qualquer placa e facilmente instalar todas as suas dependências, conforme mostra a Figura 25. Foi pesquisado pela placa apropriada (ESP32 DEVKIT) e foi selecionado da lista a placa produzida pela *DOIT* por ser uma placa de modelo bastante próximo a placa ESP32 DevKitC V4 utilizada.

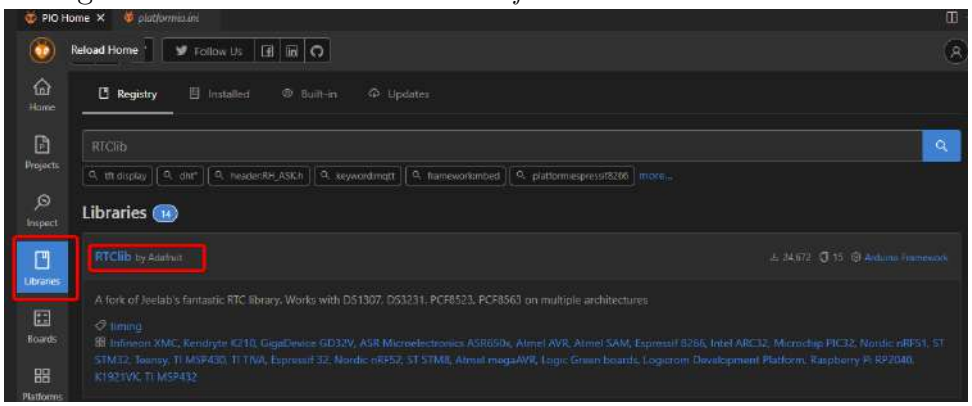
Figura 25 – Menu *Boards* do *PlatformIO*



Fonte: Próprio autor.

Feito isso, é necessário apenas criar um novo projeto a partir do menu da página inicial mostrado na Figura 24, e a partir desse menu selecionar a placa que foi instalada como base para o projeto, além do framework utilizado. Para esse projeto, foi escolhido o framework Arduino que funciona com placas ESP32, como mostra a figura abaixo.

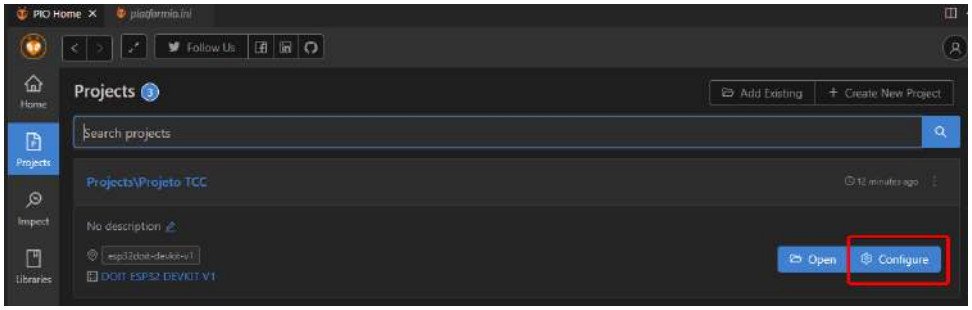
Figura 26 – Aba *Libraries* do *PlatformIO*



Fonte: Próprio autor.

O novo projeto é criado conforme as configurações especificadas. Para a inclusão de bibliotecas, elas podem ser baixadas pelo menu *Libraries* de forma similar a instalação das dependências da placa. Para esse projeto, são utilizadas as bibliotecas: *RTClib* e *BusIO* da *Adafruit*, *ThingSpeak* da *Mathworks* e a biblioteca *ModbusMaster*, como mostra a figura abaixo.

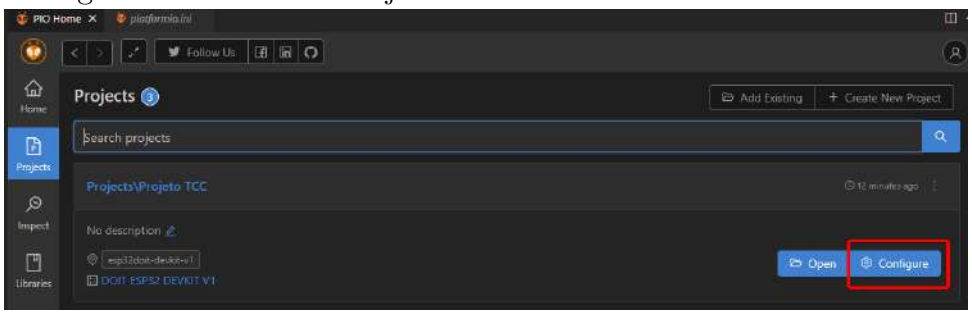
Figura 27 – Lista de Projetos



Fonte: Próprio autor.

Para a inclusão das bibliotecas no projeto, acessamos a aba *Projects* (figura 28) e lá encontramos o projeto criado. Acessamos as configurações clicando em *Configure*.

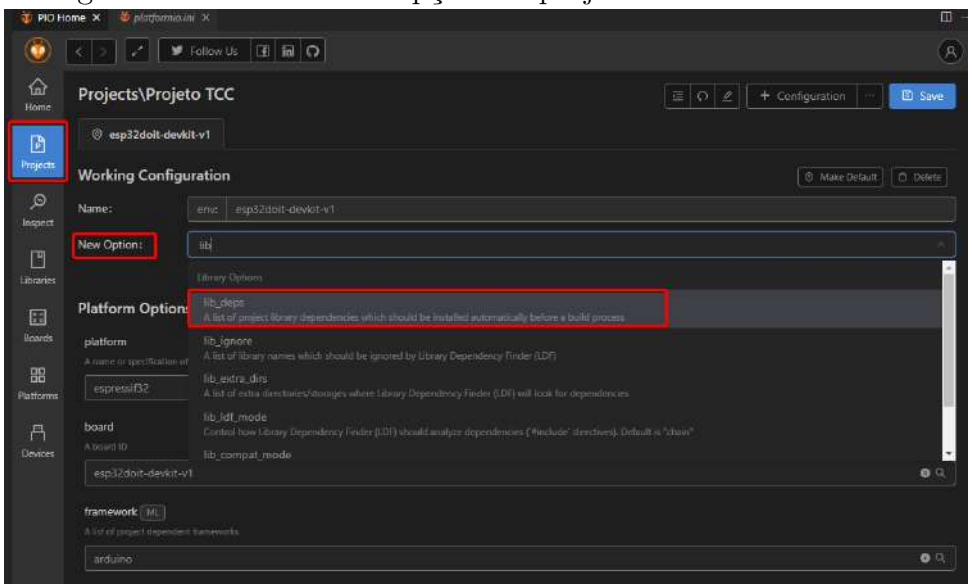
Figura 28 – Lista de Projetos



Fonte: Próprio autor.

Lá podemos encontrar as opções de projeto, e adicionar novas opções, como mostra a figura abaixo.

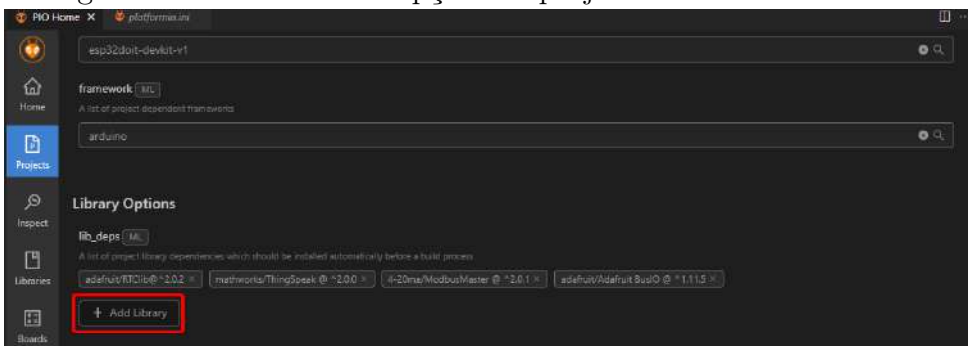
Figura 29 – Adicionando opções de projeto



Fonte: Próprio autor.

Para esse projeto, adicionaremos a opção *lib deps*, e com essa opção adicionada, podemos adicionar as bibliotecas clicando em *Add Library* e selecionando as bibliotecas.

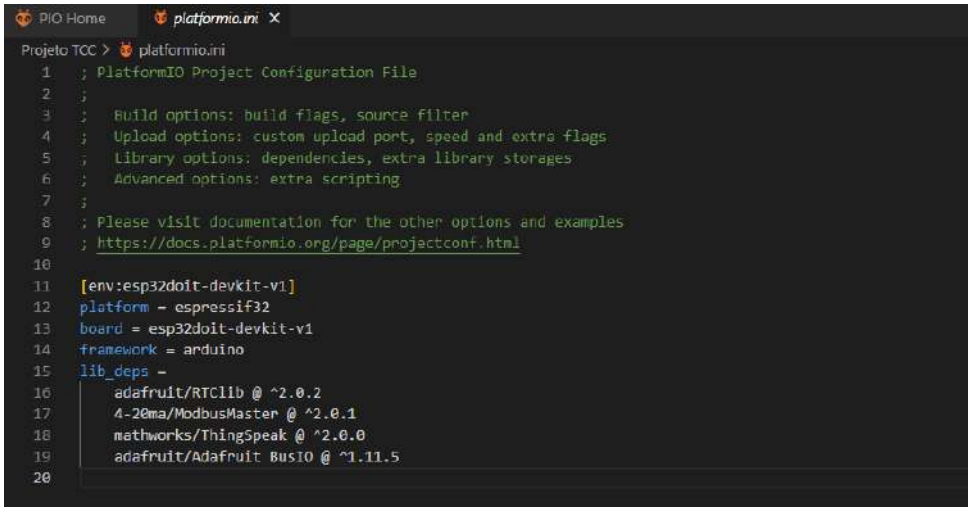
Figura 30 – Adicionando opções de projeto



Fonte: Próprio autor.

O projeto está agora devidamente criado e configurado. Após a criação e configuração, são criados automaticamente dois arquivos principais para o projeto: O *platformio.ini* é o arquivo onde são armazenadas as configurações do projeto. É recomendável interagir com as configurações apenas pelo menu do *PlatformIO*, mas caso necessário, é possível realizar mudanças diretamente no arquivo de configurações.

Figura 31 – Arquivo *platformio.ini*

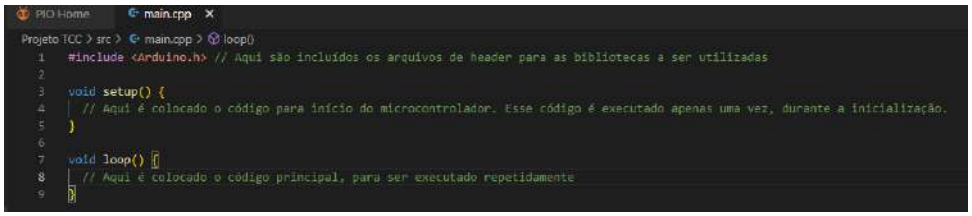


Fonte: Próprio autor.

O outro arquivo principal é o *main.cpp*, o arquivo onde o código do programa a ser executado pelo microcontrolador é realizado. Conforme a extensão do arquivo mostra, é um arquivo para programação em C++, linguagem especializada em programação orientada a objetos. Ao ser criado, o arquivo inclui apenas código para a inclusão da biblioteca do framework principal (no caso, Arduino) e *templates* para o código de *setup*

e para o código de execução principal *loop*, que será executado repetidamente. A figura abaixo mostra o arquivo *main.cpp* logo após sua criação.

Figura 32 – Arquivo *main.cpp*

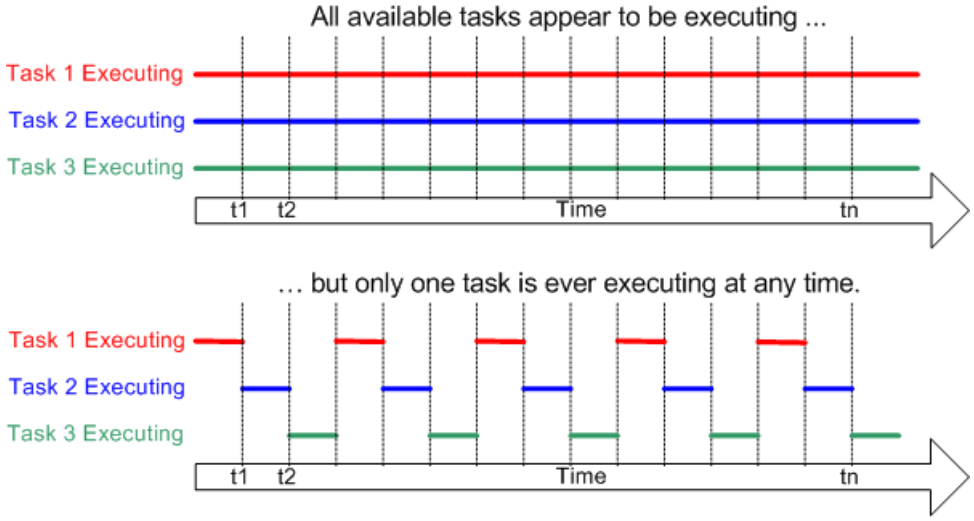


Fonte: Próprio autor.

3.2.2 FreeRTOS

Um *RTOS* (*Real-Time Operating System*) é um sistema operacional capaz de "executar"diversas tarefas ao mesmo tempo. Na realidade, um núcleo de processamento é capaz de executar apenas uma tarefa por vez. Um *RTOS* é capaz de alternar entre tarefas rapidamente, dando a impressão de que múltiplos programas estão sendo executados ao mesmo tempo em um só núcleo de processamento (FreeRTOS, a). Esse funcionamento pode ser visto na figura 33.

Figura 33 – Como funciona o *Multitasking* realizado por um *RTOS*



Fonte: (FreeRTOS, a)

Existem diversas vantagens na utilização de um *RTOS* para a escrita de software para sistemas embarcados, sendo as mais importantes para o nosso projeto (FreeRTOS, b):

- Abstração de informação de agendamento: Um dos principais componentes de um *RTOS* é o agendador em tempo real. Esse é capaz de especificar e controlar as características de temporização de um programa de forma muito mais simples e compacta.
- Manutenção e extensibilidade: A retirada das informações de temporização de dentro do código principal faz com que a mudança ou adição de algum programa ou tarefa a ser executada seja muito mais simples e não afete o funcionamento das outras tarefas, tornando o código mais simples de se manter ou de ser extendido.
- Modularidade: Um *RTOS* permite que tarefas sejam criadas de forma independente, fazendo com que não haja a necessidade de haver controles entre tarefas ou evitar que funções ou partes do programa demorem demais para que outras tarefas tenham tempo de ser executadas.

O *FreeRTOS* é um *RTOS* disponível gratuitamente pela *MIT License* de *software open-source*, feito para microcontroladores e microprocessadores e com foco na facilidade de uso e na confiabilidade. Na lógica do FreeRTOS usada no projeto, são criadas tarefas individuais para cada objetivo a ser cumprido (Exemplo: Leitura de um sensor, upload de variáveis para o *ThingSpeak* etc.). Utilizando as vantagens do *FreeRTOS*, cada tarefa pode ser controlada individualmente com seus próprios intervalos de tempo, e podem também ser criadas dependências entre tarefas, criando por exemplo tarefas que desabilitam outras tarefas temporariamente. A lógica do sistema em tempo real será detalhada mais a frente.

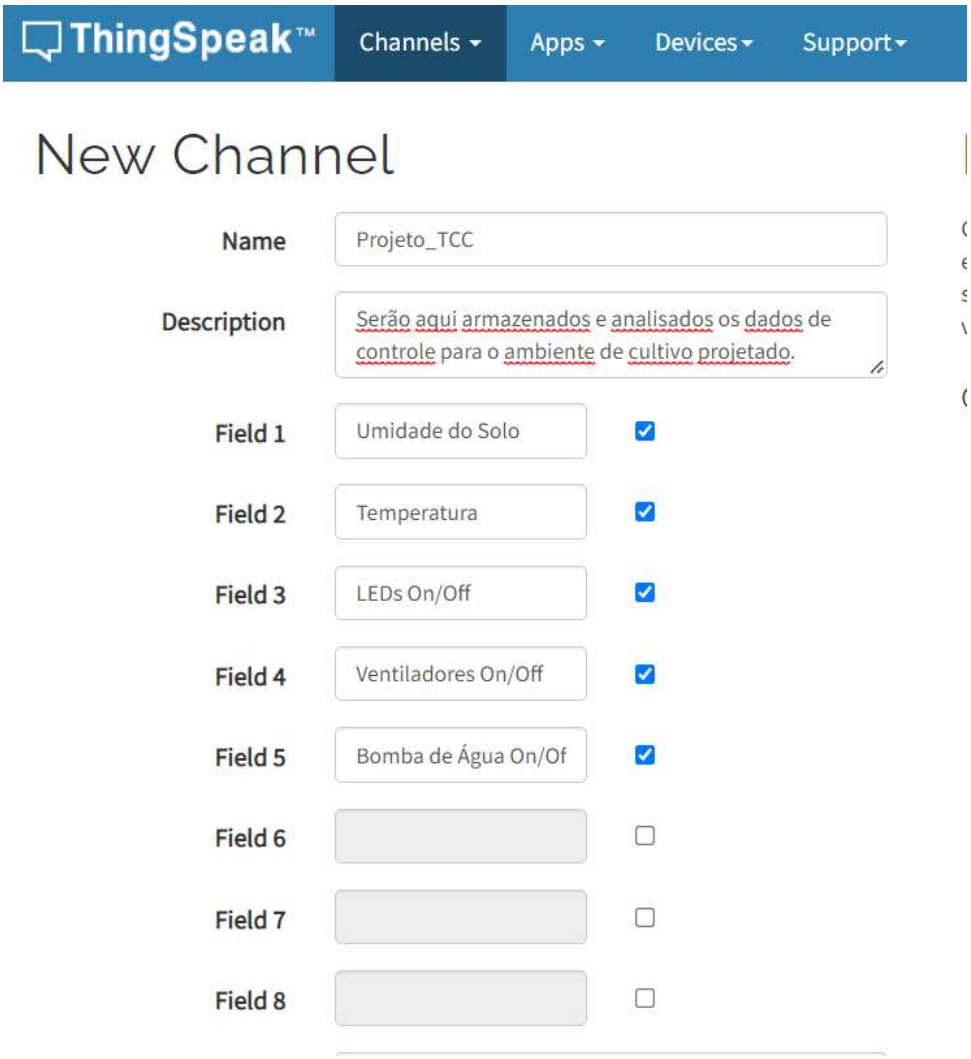
3.2.3 ThingSpeak

O *ThingSpeak* é uma ferramenta de analítica de IoT que permite visualizar e analisar dados em tempo real. Esses dados podem ser enviados de dispositivos IoT conectados ao serviço. O ThingSpeak é uma forma de analisar os dados de protótipos IoT sem que seja necessário a criação de servidores ou *software* de *web* específico para o projeto (MathWorks).

Para a utilização do serviço do *ThingSpeak* no projeto, é necessário uma conta na *MathWorks*. O *ThingSpeak* oferece a possibilidade criação de canais de comunicação para que sejam utilizados por dispositivos que tem a capacidade de conexão com a API

da plataforma. Um canal do *ThingSpeak* é criado e esse canal pode receber informações para diversos campos. Para o projeto, os seguintes campos foram criados:

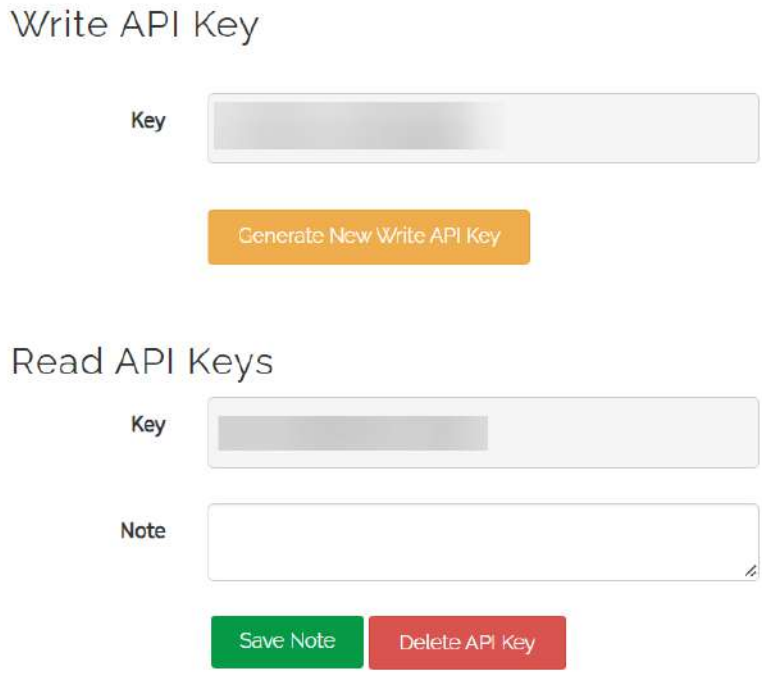
Figura 34 – Criação de canal no *ThingSpeak*



Fonte: Próprio autor.

Para a comunicação entre o canal criado e qualquer dispositivo (no caso do projeto, o microcontrolador ESP32), são utilizadas as chaves de escrita e de leitura do API do canal criado, conforme a Figura 35.

Figura 35 – Chaves do API do canal do *ThingSpeak* (borrado por segurança)



Fonte: Próprio autor.

Essas informações serão utilizadas no código para o envio das informações do projeto em tempo real.

3.3 Montagem

Nessa seção, será descrito como foi feita a montagem do protótipo do projeto, mostrando os esquemáticos e localização física dos componentes na instalação.

3.3.1 Circuito Lógico

O circuito lógico foi feito em uma protoboard, contendo os componentes lógicos necessários para o funcionamento do protótipo mencionados anteriormente, sendo eles: ESP32 (microcontrolador), módulo RTC DS3231, módulo RS485 para o sensor de temperatura e sensor de umidade do solo. A alimentação do microcontrolador ocorre por entrada microUSB conectada a um transformador de carregador de celular, com entrada da rede de 220V AC e saída 5V.

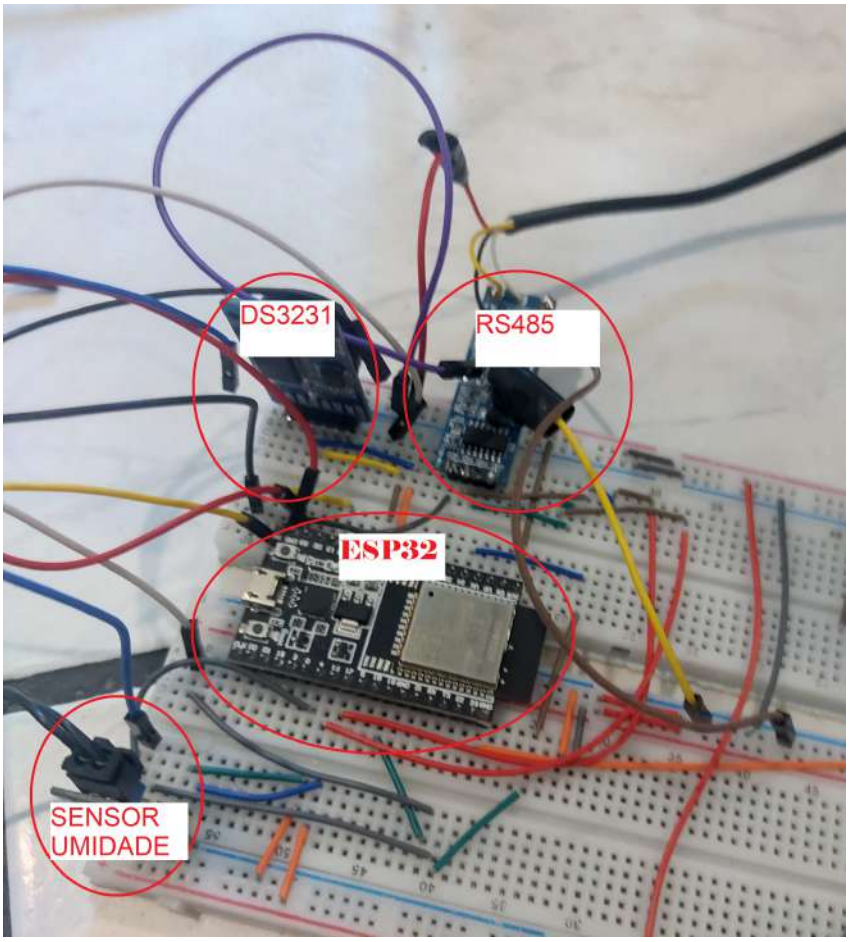
O microcontrolador possui saídas de 3.3V e de 5V, com GND comum. Para cada componente, a alimentação deve ser específica, sendo elas:

- Módulo RTC: 3.3V
- Sensor de umidade do solo: 3.3V
- Sensor de Temperatura: 3.3V
- Módulo RS485: 5V
- Módulo de relés (Não conectado na mesma protobard do circuito Lógico): 5V

As saídas para os pinos do Módulo de Relés são realizadas por jumpers do tipo macho-fêmea, conectando os pinos de saída do ESP32 da protoboard para os pinos tipo macho de entrada do relé, que podem ser vistos na figura 23.

A montagem do circuito lógico pode ser vista na figura 36.

Figura 36 – Montagem do Circuito Lógico



Fonte: Próprio autor.

Os seguintes pinos (podem ser vistos na figura 10) além das saídas de 3.3V, 5V e GND são utilizados no projeto:

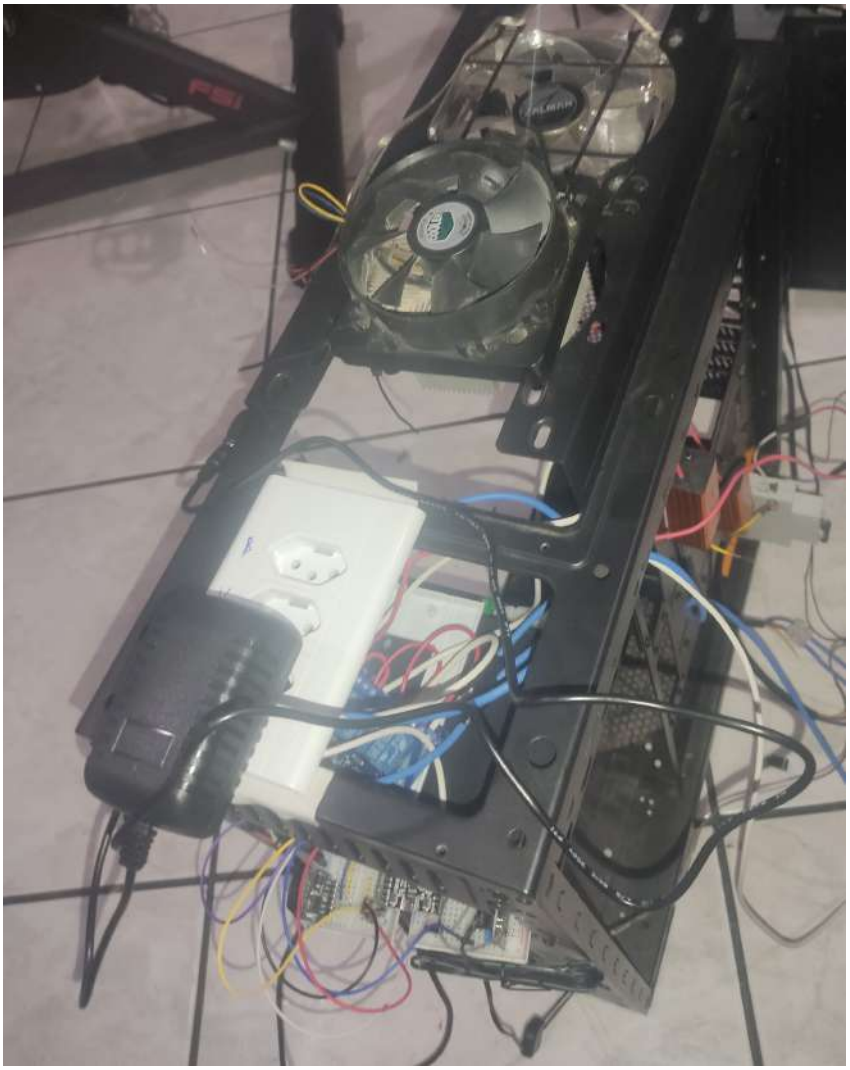
- *GPIO4*: Saída lógica de controle dos LEDs para o relé 1

- *GPIO13*: Saída lógica de controle da bomba para o relé 2
- *GPIO19*: Saída lógica de controle dos ventiladores para o relé 3
- *RXD2*: Saída RX para a comunicação *Modbus* com o sensor de temperatura
- *TXD2*: Saída TX para a comunicação *Modbus* com o sensor de temperatura
- *GPIO34*: Entrada analógica de leitura do ADC para o sensor de umidade do solo
- *RTC GPIO 17*: Conectada ao pino SDA do módulo RTC
- *RTC GPIO 16*: Conectada ao pino SCL do modulo RTC

3.3.2 Recipiente

O recipiente ou ambiente utilizado para a montagem do protótipo foi um gabinete de computador ATX, de dimensões 205x465x515mm. Todos os componentes atuadores, bem como a parte de controle, ficam situados dentro do gabinete, mostrado na figura abaixo.

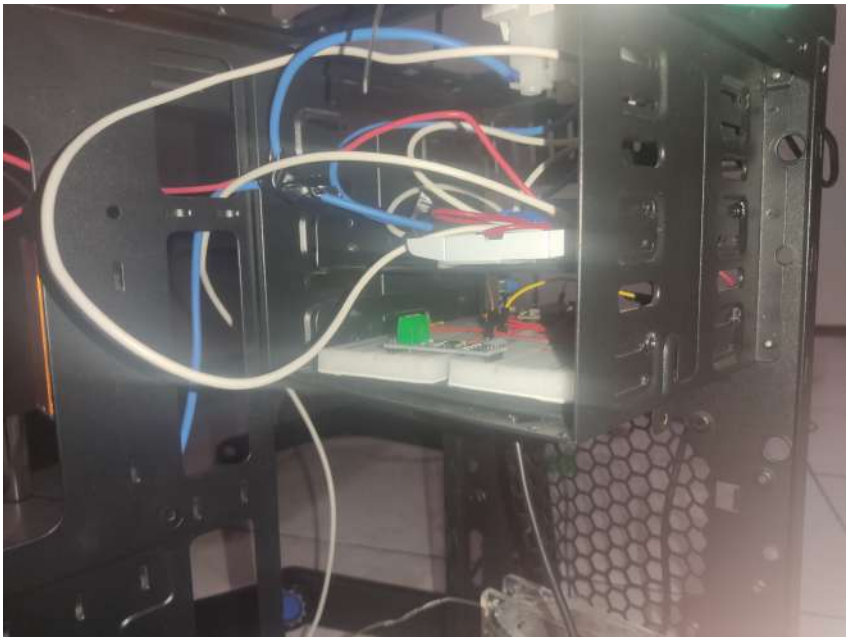
Figura 37 – Protótipo montado e aberto



Fonte: Próprio autor.

O circuito lógico juntamente com o módulo de relés ficam em um compartimento na parte de cima do gabinete, conforme a figura abaixo.

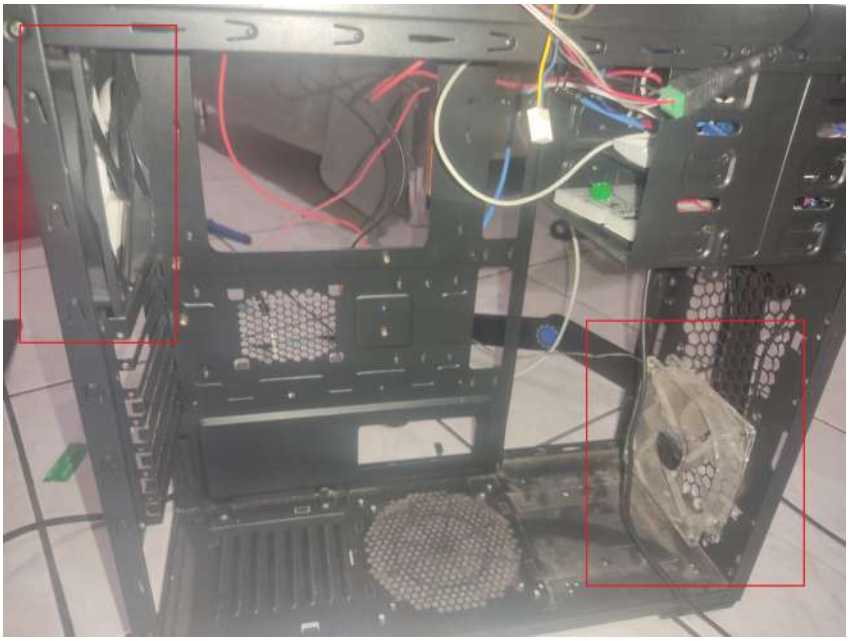
Figura 38 – Compartimento do circuito lógico



Fonte: Próprio autor.

Os ventiladores estão localizados em ambos os lados do gabinete, gerando fluxo de ar para a planta no interior e regulando a temperatura. A localização de montagem dos ventiladores pode ser vista abaixo:

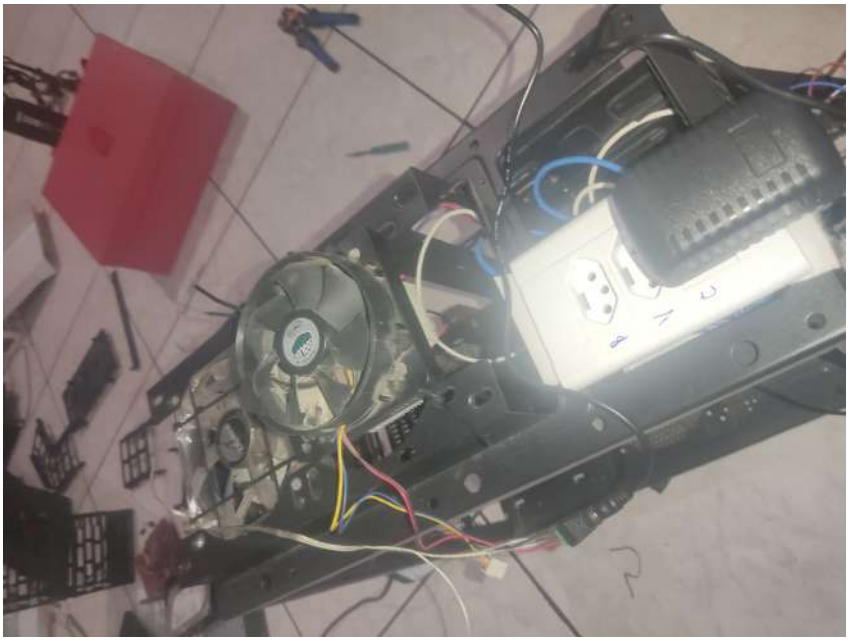
Figura 39 – Localização dos ventiladores



Fonte: Próprio autor.

As saídas do módulo de relés estão conectadas a uma tomada tripla para a utilização com os atuadores, como pode ser visto na figura abaixo.

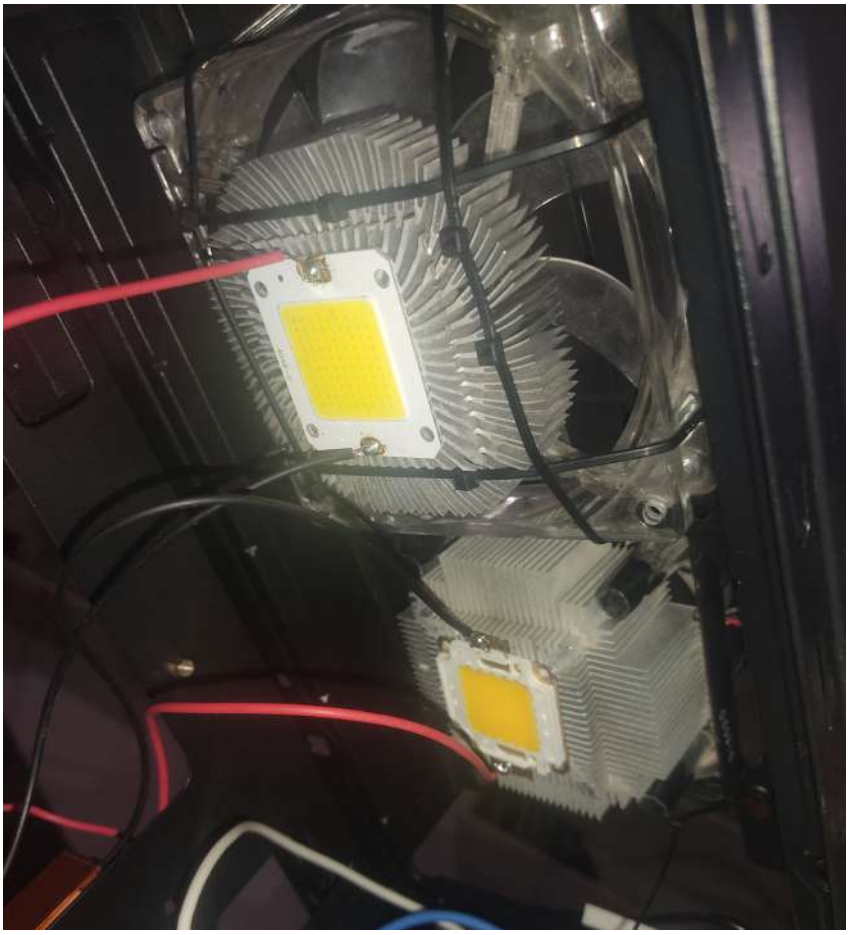
Figura 40 – Tomadas dos atuadores



Fonte: Próprio autor.

Os LEDs foram montados na parte superior do gabinete, acoplados a dissipadores de calor e ventiladores apropriados para a refrigeração, já que esse tipo de LED costuma gerar muito calor e caso não refrigerado adequadamente, pode vir a sobreaquecer e queimar. Essa montagem pode ser vista na figura abaixo.

Figura 41 – Localização dos LEDs



Fonte: Próprio autor.

A parte de tensão de rede para a alimentação do comum dos relés é controlada por um disjuntor de 10A localizado na parte traseira do gabinete, juntamente com os drivers dos LEDs.

Figura 42 – Parte traseira e localização do disjuntor

Fonte: Próprio autor.

3.4 Código e lógica de funcionamento

O código utiliza do sistema *FreeRTOS* para dividir em tarefas modulares as ações do microcontrolador. No início do processo de *boot* do ESP32, é realizado o código na função *setup()*, onde são feitas as configurações dos pinos de saída, a comunicação com o módulo RTC e o módulo RS485 com o protocolo Modbus, bem como a criação das tarefas que realizarão os processos de controle.

Após o setup, as tarefas dos sensores são iniciadas, sendo as leituras do sensor de umidade do solo realizadas a cada 10 segundos e as leituras do sensor temperatura realizadas a cada 4 segundos. A cada 30 segundos, o processo de controle (tarefa *ReceiveData()* e de upload das variáveis para o ambiente de controle do ThingSpeak é iniciado, suspendendo temporariamente o funcionamento dos sensores. Sendo assim, a cada 30 segundos ocorre um ciclo de verificação das variáveis e de controle dos atuadores. As leituras ocorrem a cada 10 segundos e a cada 4 segundos para que ocorram múltiplas leituras por ciclo e assim a possibilidade de erros de leitura provocando erros no controle seja minimizada.

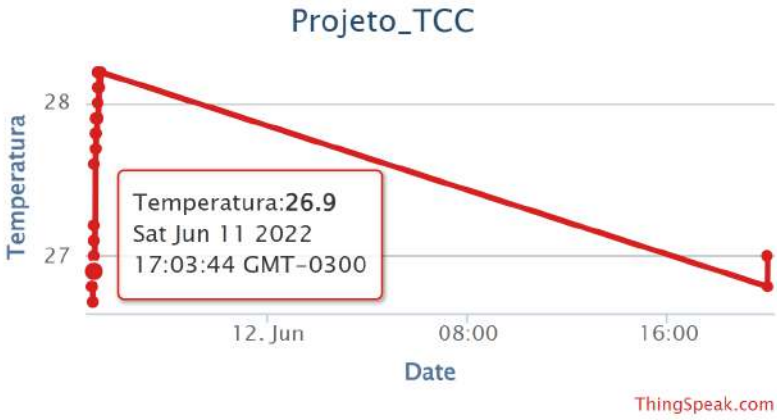
As tarefas de controle realizam a leitura das variáveis dos sensores e, baseado nessas leituras, modificam a saída dos pinos que controlam os relés de saída para os atuadores. Cada controle é realizado de forma específica de acordo com as leituras dos

sensores:

- Controle da bomba: Compara a leitura com os valores das variáveis de umidade mínima (*umidMin*) e umidade máxima (*umidMax*). Caso a umidade seja menor que a umidade mínima, liga a bomba por um período máximo de 2 minutos. Caso os 2 minutos do período máximo se passem ou a umidade atinja o nível de *umidMax*, a bomba é desligada e deve permanecer desligada por pelo menos 2 minutos.
- Controle dos ventiladores: Compara a leitura com os valores das variáveis de temperatura máxima (*tempMax*) e temperatura mínima (*tempMin*) Caso a temperatura seja maior que a temperatura máxima, liga os ventiladores por um período máximo de 5 minutos. Caso os 5 minutos do período máximo se passem ou a temperatura atinja a temperatura mínima, o ventilador é desligado e deve permanecer desligado por pelo menos 1 minuto.
- Controle dos LEDs: Verifica a hora do dia. Caso a hora esteja dentro do intervalo definido pelas variáveis *horaMin* e *horaMax*, os LEDs ficarão ligados.

Ao fim do ciclo de controle, uma última tarefa é executada, enviando as variáveis de controle como as leituras da temperatura e da umidade do solo, bem como os estados de cada atuador para a plataforma ThingSpeak, onde os resultados poderão ser observados. Ao fim dessa tarefa, as tarefas de leitura dos sensores voltam a ser executadas, iniciando assim o novo processo de leituras e a espera para o próximo ciclo de controle.

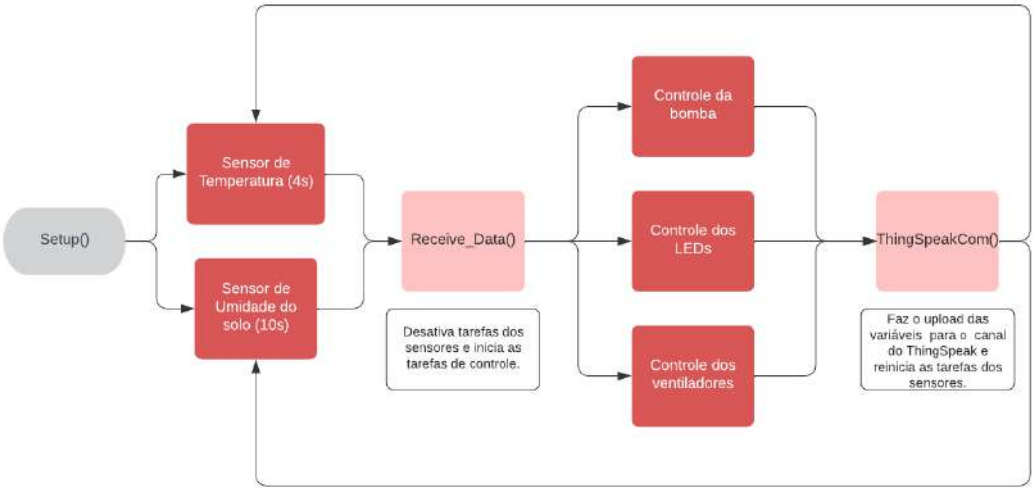
Figura 43 – Exemplo de gráfico de leitura da temperatura no ThingSpeak



Fonte: Próprio autor.

O fluxograma resumido do funcionamento do programa pode ser visto na figura 47.

Figura 44 – Fluxograma de funcionamento para o projeto



Fonte: Próprio autor.

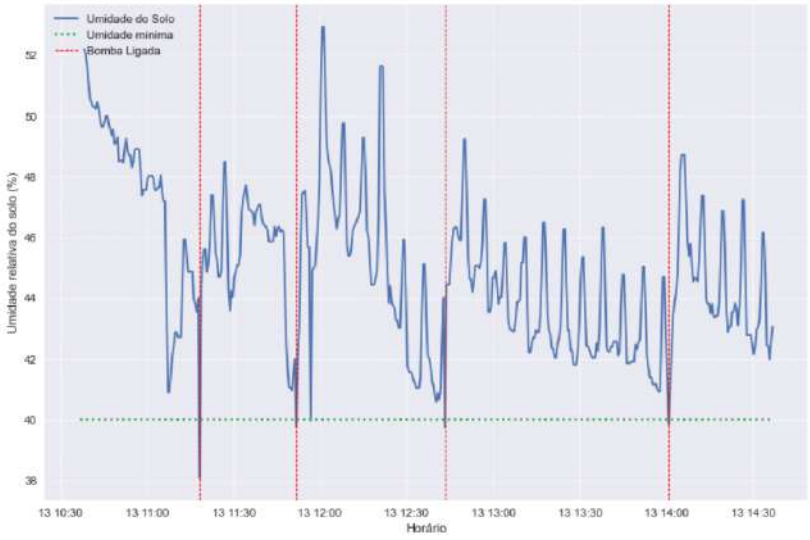
4 RESULTADOS

Neste capítulo, iremos apresentar os dados experimentais obtidos, partindo da visão do controle de cada uma das variáveis alvo: temperatura, umidade do solo e iluminação ou fotoperíodo

4.1 Umidade do Solo

O sistema de controle da umidade do solo, conforme visto na seção 3.4 funciona com base em um valor de umidade mínima e no funcionamento da bomba por no máximo 2 minutos, ou caso a umidade atinja um valor de umidade máximo. Para os testes, o valor de umidade mínima escolhido foi de 40% de umidade relativa, e a umidade máxima foi de 60%.

Figura 45 – Funcionamento do controle de umidade do solo (4 horas de duração)



Fonte: Próprio autor.

Podemos perceber, pela figura 45 que o sistema atuou nos momentos onde a leitura do sensor foi abaixo de 40% de umidade, e a umidade voltava a subir assim que possível. Apesar de não atingir a umidade máxima de 60%, o sistema desliga após 2 minutos, fazendo com que a umidade não suba demasiadamente, funcionando de acordo com o código formulado.

4.2 Iluminação ou Fotoperíodo

Para o teste realizado, o fotoperíodo foi configurada para ligamento dos LEDs às 11h da manhã e desligamento às 20h da noite.

Figura 46 – Funcionamento do controle de iluminação

Fonte: Próprio autor.

Observa-se na figura 46 o funcionamento exatamente conforme o programado, com o ligamento e desligamento do sistema de iluminação na hora correta, com atraso de apenas 4 minutos em ambos os casos (ligamento às 11:04 e desligamento às 20:04, ocorrido devido a atraso no RTC)

4.3 Temperatura

O controle dos ventiladores foi configurado com temperatura máxima de 29 graus celsius, sendo ele ligado durante 5 minutos quando acima dessa temperatura, desligado durante 1 minuto, e ligado novamente (caso ainda esteja acima da temperatura máxima). O gráfico do funcionamento do controle de temperatura durante o teste pode ser visto na figura abaixo.

Figura 47 – Funcionamento do controle de temperatura

Fonte: Próprio autor.

Observa-se que o ligamento dos ventiladores não é suficiente para diminuir a temperatura abaixo dos 29 graus estipulados como máximo. Apesar disso, a temperatura se mantém estável após o ligamento dos ventiladores, não aumentando demasiadamente. O sistema foi testado em ambiente fechado, sem qualquer ar-condicionado ou outro tipo de sistema de climatização, o que pode explicar esse funcionamento, já que é difícil para o sistema diminuir sua temperatura trocando calor com um ambiente quente.

Percebe-se também o aumento da temperatura ao se ligar os LEDs, bem como a diminuição quando ocorre o desligamento. Isso mostra que o sistema ainda necessita de um ambiente externo que colabore com a troca de calor para casos onde a temperatura deve ser mantida em níveis baixos.

O funcionamento dos ventiladores é importante também para as trocas de gases, já que as plantas, durante o processo de fotossíntese, utilizam gás carbônico e liberam oxigênio, mudando a concentração desses gases no sistema fechado.

5 CONCLUSÃO

O cultivo de plantas já é um passatempo popular no brasil, mas a utilização da tecnologia para a automatização de processos de cultivo ainda é praticamente inexistente e inacessível para o consumidor residencial. O objetivo do projeto foi apresentar uma proposta simples para esse tipo de tecnologia a nível doméstico.

O protótipo montado consistia em um ambiente de estufa onde seria possível controlar as variáveis mais importantes para o cultivo de plantas: iluminação, umidade do solo (regagem), temperatura e ventilação. Para isso, foi essencial o uso do microcontrolador ESP32 com ótimas características para o projeto, como a capacidade de utilização de wi-fi, assim como outros componentes periféricos como sensores e atuadores. A montagem física do projeto satisfatória, especialmente pela facilidade da alocação do espaço e das peças dentro de um gabinete de computador, que já possui entradas apropriadas para alguns dos componentes utilizados, como os ventiladores.

O funcionamento do protótipo foi satisfatório, conseguindo controlar o ambiente de maneira adequada e conforme o código do program. O controle da temperatura mostrou-se ainda dependente do ambiente externo, sendo necessário um atuador capaz de controle maior da temperatura para torná-lo totalmente independente do ambiente, como um aquecedor ou refrigerador, a depender da necessidade.

Para projetos futuros, pode ser criado uma interface de usuário, onde seria possível controlar fatores como umidade mínima, temperatura máxima e fotoperíodo de forma que essas variáveis possam ser alteradas dinamicamente. Pode também ser feito, como mencionado, o controle da temperatura de forma mais robusta a ponto de tornar o sistema totalmente independente de fatores externos.

REFERÊNCIAS

AMABIS, MARTHO: *Biologia dos Organismos*. Bd. 2. 3. Moderna, 2009. – ISBN 978-85-16-07417-3

ANTUNES ET AL., Luis Eduardo C.: *Morangueiro*. EMBRAPA, 2016. – ISBN 978-85-7035-594-2

BARNITZ: *Growing with Artificial Light*. 2022. – URL <https://www.bobsmarket.com/blog/growing-with-artificial-light>

BLAKEMORE, Erin: *What was the Neolithic Revolution?* 2019. – URL <https://www.nationalgeographic.com/culture/article/neolithic-agricultural-revolution>

CCL: *WHY YOU SHOULD NOT USE COOL WHITE LEDS BEHIND COLOURED LENSES*. 2018. – URL <https://www.classiccarleds.co.uk/blogs/news/why-you-should-not-use-white-leds-behind-coloured-lenses>

ESPRESSIF: *ESP32 Series Datasheet*. 3,9. : Espressif Systems (Veranst.)

FREERTOS: *What is An RTOS?*. – URL <https://www.freertos.org/about-RTOS.html>

FREERTOS: *Why use an RTOS*. – URL <https://www.freertos.org/FAQWhat.html#WhyUseRTOS>

GOOGLE: *Google Trends*. 2022. – URL <trends.google.com>

HARDIGREE, Peggy A.: *The edible indoor garden*. St. Martin's Press, 1980. – ISBN 0-312-23689-1

HISTORY, Editors: *Hunter-Gatherers*. 2018. – URL <https://www.history.com/topics/pre-history/hunter-gatherers>

INTEGRATED, Maxim: *DS3231 Datasheet*. 2015. – URL <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>

MATHWORKS: *ThingSpeak Documentation*. – URL <https://www.mathworks.com/help/thingspeak/>

MISCHIANTI, Renzo: *ESP32 DevKitC v4 high resolution pinout and specs*. 2021. – URL <https://www.mischianti.org/2021/07/17/esp32-devkitc-v4-high-resolution-pinout-and-specs/>

MOREIRA, Rosa: *Deficiência de nutrientes nas plantas: sabe quais são os principais sintomas?* 2018. – URL <https://agriculturaemar.com/deficiencia-de-nutrientes-nas-plantas-sabe-quais-sao-os-principais-sintomas/>

NISE, Norman S.: *Engenharia de Sistemas de Controle*. 6. LTC, 2013. – ISBN 978-85-216-2136-2

ROPELLI, Marco V.: *Jardinagem é terapia caseira na quarentena*. 2020. – URL <https://www.imparcial.com.br/noticias/jardinagem-e-terapia-caseira-na-quarentena,37630>

RUEHR, Thomas A.: *How vegetable plant roots absorb nutrients*. 2018. – URL <https://www.cropnutrition.com/resource-library/how-vegetable-plant-roots-absorb-nutrients>

RYE, JURUKOVSKI DESAIX AVISSAR C.: *Biology*. OpenStax, 2016

Sensirion: *SHT 20 Datasheet*. 2014. – URL <https://sensirion.com/media/documents/CCDE1377/617A8CB7/Sensirion_Humidity_Sensors_SHT20_Datasheet.pdf>

VERNIER: *What Are the Best Light Sources For Photosynthesis*. 2018. – URL <https://www.vernier.com/2018/09/04/what-are-the-best-light-sources-for-photosynthesis/>

APÊNDICE A – CÓDIGO FONTE

```
12pt#include <Arduino.h>
#include <ModbusMaster.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "freertos/semphr.h"
#include "WiFi.h"

#include <SPI.h>
#include <Wire.h>
#include "RTClib.h"

#include <ThingSpeak.h>

#define UMID_SOIL_PIN 34
#define LIGHT_PIN 4
#define BOMBA_PIN 13
#define VENT_PIN 19
#define COOLER_PIN 26

#define RXD2 16 //DI
#define TXD2 17 //RO
#define MODBUS_DATA_TRANSACTION_PIN 8


ModbusMaster node; // objeto do ModbusMaster
uint8_t Slave_ID = 1; // ID do sensor slave no protocolo Modbus

// Tasks and Queues
TaskHandle_t xKeepAliveHandle;
```

```
31 TaskHandle_t xinitWiFiHandle;
32 TaskHandle_t sensor_solo_handle;
33 TaskHandle_t sensor_temp_handle;
34 TaskHandle_t receive_data_handle;
35
36
37
38 QueueHandle_t xFilaH;
39 QueueHandle_t xFilaT;
40
41 char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terça", "Quarta",
   ↪  "Quinta", "Sexta", "Sábado"};
42 RTC_DS3231 RTC;
43
44 // Wifi declarations
45 void initWiFi(void *pvParameters);
46 void keepWiFiAlive(void *pvParameters);
47 //Sensor Declarations
48 void sensor_solo(void *pvParameters);
49 void sensor_temp(void *pvParameters);
50 void receive_data(void *pvParameters);
51
52 //Control Declarations
53 void control_bomba(void *pvParameters);
54 void control_led(void *pvParameters);
55 void control_cooler(void *pvParameters);
56 int flagBomba = 0;
57 int flagLED = 0;
58 int flagCooler = 0;
59
60 TaskHandle_t control_bomba_handle;
61 TaskHandle_t control_led_handle;
```

```
62 TaskHandle_t control_cooler_handle;
63
64 WiFiClient client;
65
66 //Thingspeak Declarations
67 const long CHANNEL = ; //Removido por segurança pelo autor
68 const char *WRITE_API = ""; //Removido por segurança pelo autor
69 void ts_com(void *pvParameters);
70 TaskHandle_t ts_com_handle;
71
72 void initWiFi(void *pvParameters)
73 {
74   const char *ssid = ""; //Nome da rede wi-fi; Removido por segurança
     ↪  pelo autor
75   const char *password = "sigerregis"; //Senha da rede wi-fi; Removido
     ↪  por segurança pelo autor
76   unsigned long previousMillis = 0;
77
78   while (1)
79   {
80     unsigned long interval = 5000;
81     unsigned long currentMillis = millis();
82
83     Serial.println("Iniciada initWiFi \n");
84     WiFi.mode(WIFI_STA);
85     WiFi.begin(ssid, password);
86     Serial.print("Connecting to WiFi ..");
87     while ((WiFi.status() != WL_CONNECTED))
88     {
89       Serial.print('.');
90       currentMillis = millis();
91       vTaskDelay(pdMS_TO_TICKS(1000));
```

```
92
93        if ((currentMillis - previousMillis >= interval))
94        {
95
96          previousMillis = currentMillis;
97
98          // xTaskNotifyGive(leituraHandle);
99
100         break;
101
102       }
103     }
104     if (WiFi.status() == WL_CONNECTED)
105     {
106       Serial.println(WiFi.localIP());
107       Serial.println("Task Suspendida \n");
108       vTaskResume(xKeepAliveHandle);
109       vTaskDelay(pdMS_TO_TICKS(500));
110       vTaskSuspend(NULL);
111     }
112   }
113 }
114
115 void keepWiFiAlive(void *pvParameters)
116 {
117
118   unsigned long previousMillis = 0;
119
120   while (1)
121   {
122
123     unsigned long interval = 5000;
```

```
124     unsigned long currentMillis = millis();
125
126     // Serial.println("Iniciada keepAlive\n");
127     if ((WiFi.status() != WL_CONNECTED) && (currentMillis -
        ↪  previousMillis >= interval))
128     {
129
130       WiFi.disconnect();
131       WiFi.reconnect();
132       previousMillis = currentMillis;
133       Serial.println(WiFi.localIP());
134     }
135
136     // Serial.println(String(previousMillis));
137     // Serial.println(String(currentMillis));
138     Serial.println(WiFi.localIP());
139
140     // xTaskNotifyGive(leituraHandle);
141     vTaskDelay(pdMS_TO_TICKS(20000));
142   }
143 }
144
145 void ts_com(void *pvParameters)
146 {
147   float humRecebida, setTempRecebida, setUmidRecebida;
148   // int tempRecebida = 100;
149   float soil_umid;
150   float temp;
151
152   while(1)
153   {
154     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
```

```
155     if ((WiFi.status() == WL_CONNECTED))
156     {
157       xQueueReceive(xFilaH, &soil_umid, pdMS_TO_TICKS(10000));//recebe a
        ↪  leitura da umidade
158       xQueueReceive(xFilaT, &temp, pdMS_TO_TICKS(10000));//recebe a
        ↪  leitura de temperatura
159       //Estabelece os valores a ser enviados
160       ThingSpeak.setField(1, soil_umid);
161       ThingSpeak.setField(2, temp);
162       ThingSpeak.setField(3, flagLED);
163       ThingSpeak.setField(4, flagCooler);
164       ThingSpeak.setField(5, flagBomba);
165       // Escreve no canal do ThingSpeak
166       int x = ThingSpeak.writeFields(CHANNEL, WRITE_API); //envia os
        ↪  valores para os campos específicos
167       if (x == 200)
168       {
169         Serial.println("Update concluido");
170       }
171       else
172       {
173         Serial.println("Erro HTTP: " + String(x));
174       }
175     }
176     vTaskResume(sensor_solo_handle);
177     vTaskResume(sensor_temp_handle);
178     vTaskDelay(pdMS_TO_TICKS(100));
179   }
180 }
181
182 void receive_data(void *pvParameters){
183   while(1)
```

```
184   {
185     if (millis() > 50000)
186     { //só inicia a tarefa depois de 50 segundos do microcontrolador
        ↪  ligado
187       vTaskSuspend(sensor_solo_handle); //suspendendo a tarefa do sensor
          ↪  de umidade
188       vTaskSuspend(sensor_temp_handle); //suspendendo a tarefa do sensor
          ↪  de temperatura
189       Serial.println("Iniciando ciclo de controle...");
190       xTaskNotifyGive(control_bomba_handle); //Sinalizando a tarefa de
          ↪  controle para iniciar
191       //vTaskResume(sensor_solo_handle);
192       vTaskDelay(pdMS_TO_TICKS(30000));
193     }
194   }
195
196 }
197
198 void control_bomba(void *pvParameters){
199   float umid = 0;
200   float umidMin = 40;
201   float umidMax = 60;
202   boolean bomba = HIGH;
203
204   // variaveis de controle
205   unsigned long Bomba_Min_Off = 2 * 60 * 1000;
206   unsigned long Bomba_Max_On = 2 * 60 * 1000;//2 * 60 * 1000;
207   // vars de tempo
208   unsigned long timeOFF = 0;
209   unsigned long timedummy = 0;
210   unsigned long timeON = 0;
211   digitalWrite(BOMBA_PIN, bomba);
```

```
212
213   while(1)
214   {
215     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
216     xQueueReceive(xFilaH, &umid, portMAX_DELAY); //recebe a variável de
      ↪  leitura do sensor de umidade
217     if (bomba == HIGH){ //caso: bomba desligada
218       timedummy = millis() - timeOFF;
219       if(timedummy > Bomba_Min_Off && umid < umidMin){
220         timeON = millis();
221         flagBomba = 1;
222         bomba = LOW;
223         digitalWrite(BOMBA_PIN, bomba); //liga o pino de saída conectado
          ↪  ao relé
224         Serial.println("\nBomba ligada\n");
225       } //
226     }
227     else{ //caso: bomba ligada
228        timedummy = millis() - timeON;
229        if(timedummy > Bomba_Max_On || umid > umidMax){
230         timeOFF = millis();
231         flagBomba = 0;
232         bomba = HIGH;
233         digitalWrite(BOMBA_PIN, bomba);//desliga o pino de saída
          ↪  conectado ao relé
234         Serial.println("\nBomba desligada\n");
235        }
236     }
237     xQueueOverwrite(xFilaH, &umid);
238     xTaskNotifyGive(control_cooler_handle);
239     vTaskDelay(pdMS_TO_TICKS(100));
240   }
```

```
241
242 }
243 void control_cooler(void *pvParameters){
244   float temp = 0;
245   float tempMin = 24;
246   float tempMax = 29;
247   float tempREC = 0;
248   boolean cooler = HIGH;
249
250   // variaveis de controle
251   unsigned long cooler_Min_Off = 1 * 60 * 1000;
252   unsigned long cooler_Max_On = 5 * 60 * 1000;
253   // vars de tempo
254   unsigned long timeOFF = 0;
255   unsigned long timedummy = 0;
256   unsigned long timeON = 0;
257   digitalWrite(COOLER_PIN, cooler);
258
259   while(1){
260     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
261     xQueueReceive(xFilaT, &tempREC, portMAX_DELAY);
262     if (tempREC > 0){
263       temp = tempREC; //caso não haja leitura válida durante o período
264       //entre dois ciclos de controle, mantem a leitura anterior
265     }
266     if (cooler == HIGH){ //caso: cooler desligada
267       timedummy = millis() - timeOFF;
268       if(timedummy > cooler_Min_Off && temp > tempMax){
269         timeON = millis();
270         flagCooler = 1;
271         cooler = LOW;
```

```
272           digitalWrite(VENT_PIN, cooler); //liga o pino de saída conectado
              ↪  ao relé
273           Serial.println("\nCooler ligado\n");
274         } //
275       }
276       else{ //caso: cooler ligada
277          timedummy = millis() - timeON;
278          if(timedummy > cooler_Max_On || temp < tempMin){
279           timeOFF = millis();
280           flagCooler = 0;
281           cooler = HIGH;
282           digitalWrite(VENT_PIN, cooler); //desliga o pino de saída
              ↪  conectado ao relé
283           Serial.println("\nCooler desligado\n");
284          }
285
286       }
287       xQueueOverwrite(xFilaT, &temp);
288       xTaskNotifyGive(control_led_handle);
289       vTaskDelay(pdMS_TO_TICKS(100));
290     }
291
292 }
293 void control_led(void *pvParameters){
294   boolean light = LOW;
295   boolean cooler = HIGH;
296   int horaMax = 20;
297   int horaMin = 11;
298   pinMode(LIGHT_PIN, OUTPUT);
299   digitalWrite(LIGHT_PIN, light);
300   digitalWrite(COOLER_PIN, cooler);
301   DateTime now = RTC.now();
```

```
302   time_t hora = 0;
303
304   while (1)
305   {
306     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
307     now = RTC.now();
308     hora = now.hour();
309
310     if ((hora >= horaMin) && (hora < horaMax)) // Se a hora estiver no
      ↪  intervalo definido a Luz ficará acessa.
311     {
312       light = HIGH;
313       cooler = LOW;
314       digitalWrite(COOLER_PIN, cooler);
315       digitalWrite(LIGHT_PIN, light);
316       Serial.println("Luz Acesa");
317       Serial.println(hora);
318       flagLED = 1;
319     }
320     else
321     { // Se a hora estiver fora do intervalo definido a Luz ficará
      ↪  apagada.
322       light = LOW;
323       cooler = HIGH;
324       digitalWrite(LIGHT_PIN, light);
325       digitalWrite(COOLER_PIN, cooler);
326       Serial.println("Luz Apagada");
327       Serial.println(hora);
328       flagLED = 0;
329     }
330     xTaskNotifyGive(ts_com_handle);
331     vTaskDelay(pdMS_TO_TICKS(100));
```

```
332   }
333 }
334
335 void sensor_solo(void *pvParameters){
336   int soil_umid = 0;
337   float umid_final;
338   int SECO_READ = 4095;
339   int UMIDO_READ = 0;
340   while(1){
341     soil_umid = analogRead(UMID_SOIL_PIN);
342     Serial.println("Reading= " + String(soil_umid));
343     umid_final = 100*(((float)SECO_READ) -
      ↪  ((float)soil_umid))/(((float)SECO_READ) - ((float)UMIDO_READ));
344     Serial.println("Umidade do Solo: " + String(umid_final) + "%");
345     xQueueOverwrite(xFilaH, &umid_final);
346     vTaskDelay(pdMS_TO_TICKS(10000));
347   }
348 }
349
350 void sensor_temp(void *pvParameters)
351 {
352   short a,b;
353   float tempEnviada, humEnviada;
354   float tempError = 0;
355
356   while (1)
357   {
358     a = node.readInputRegisters(Slave_ID, 1);
359     b = node.getResponseBuffer(0);
360     tempEnviada = b;
361     tempEnviada = tempEnviada / 10;
362     if (tempEnviada > 0){
```

```
363         xQueueOverwrite(xFilaT, &tempEnviada); //envia variável de
            ↪  temperatura para a tarefa de controle
364       }
365       else{
366         xQueueOverwrite(xFilaT, &tempError); //caso haja erro de leitura,
            ↪  envia variável com valor indicando o erro
367       }
368       node.clearResponseBuffer();
369
370       Serial.println("Temp = " + String(tempEnviada));
371       vTaskDelay(pdMS_TO_TICKS(4000));
372     }
373   }
374
375   void setup() {
376     //Pin Modes
377     pinMode(BOMBA_PIN, OUTPUT);
378     pinMode(VENT_PIN, OUTPUT);
379     pinMode(COOLER_PIN, OUTPUT);
380     digitalWrite(BOMBA_PIN, HIGH);
381     digitalWrite(VENT_PIN, HIGH);
382     digitalWrite(COOLER_PIN, HIGH);
383
384
385
386     Serial.begin(9600); // Inicializa porta serial para monitoração
387     Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2); //Inicializa porta serial
        ↪  para comunicação Modbus
388     node.begin(Slave_ID, Serial2);
389
390     Wire.begin(14, 27); //Inicializa pinos de comunicação SDA e SCL com o
        ↪  módulo RTC
```

```
391   if (! RTC.begin()) {
392      Serial.println("Não foi possível encontrar RTC");
393      while (1);
394   }
395   if(RTC.lostPower()){ //SE RTC FOI LIGADO PELA PRIMEIRA VEZ / FICOU SEM
      ↪  ENERGIA / ESGOTOU A BATERIA
396     Serial.println("DS3231 OK!"); //IMPRIME O TEXTO NO MONITOR SERIAL
397     RTC.adjust(DateTime(F(__DATE__), F(__TIME__))); //CAPTURA A DATA E
        ↪  HORA EM QUE O CÓDIGO É COMPILADO E CONFIGURA O RTC
398   }
399   delay(100);
400
401   xFilaH = xQueueCreate(1, sizeof(int)); //Umidade do Solo - Fila
402   xFilaT = xQueueCreate(1, sizeof(int));
403   xTaskCreatePinnedToCore(initWiFi, "initWiFi", configMINIMAL_STACK_SIZE
      ↪  + 5000, NULL, 1, &xinitWiFiHandle, 0); //Inicializa o WiFi do
      ↪  ESP32
404   xTaskCreatePinnedToCore(keepWiFiAlive, "keepWiFiAlive", 5000, NULL, 1,
      ↪  &xKeepAliveHandle, 0);
405   // Data Treatment
406
407   xTaskCreate(sensor_solo, "sensor_solo", configMINIMAL_STACK_SIZE +
      ↪  5000, NULL, 3, &sensor_solo_handle);
408   xTaskCreate(sensor_temp, "sensor_temo", configMINIMAL_STACK_SIZE +
      ↪  5000, NULL, 2, &sensor_temp_handle);
409   xTaskCreate(control_led, "control_bomba", configMINIMAL_STACK_SIZE +
      ↪  5000, NULL, 4, &control_led_handle);
410   xTaskCreate(control_cooler, "control_cooler", configMINIMAL_STACK_SIZE
      ↪  + 5000, NULL, 5, &control_cooler_handle);
411   xTaskCreate(control_bomba, "control_bomba", configMINIMAL_STACK_SIZE +
      ↪  5000, NULL, 6, &control_bomba_handle);
```

```
412   xTaskCreate(receive_data, "receive_data", configMINIMAL_STACK_SIZE +
      ↪  5000, NULL, 1, &receive_data_handle);
413   xTaskCreate(ts_com, "ts_com", configMINIMAL_STACK_SIZE + 5000, NULL, 7,
      ↪  &ts_com_handle);
414
415   ThingSpeak.begin(client);
416   vTaskSuspend(xKeepAliveHandle);
417 }
418
419
420
421 void loop() {
422 vTaskDelay(pdMS_TO_TICKS(500));
423 }
```