FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ALLAN CARLOS DE SOUSA
ALEXANDRE LOPES DA SILVA BENEDITO
JAIRO FLORENCIO SANTOS

AMBIENTE COM TEMPERATURA E UMIDADE CONTROLADA PARA SALAS SMT POR MEIO DA IOT

São Bernardo do Campo - SP
Junho/2023

ALLAN CARLOS DE SOUSA
ALEXANDRE LOPES DA SILVA BENEDITO
JAIRO FLORENCIO SANTOS

AMBIENTE COM TEMPERATURA E UMIDADE CONTROLADA PARA SALAS SMT POR MEIO DA IOT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Orientador: Professor Esp. Jorge Luis Sarapka

São Bernardo do Campo - SP
Junho/2023

ALLAN CARLOS DE SOUSA
ALEXANDRE LOPES DA SILVA BENEDITO
JAIRO FLORENCIO SANTOS

AMBIENTE COM TEMPERATURA E UMIDADE CONTROLADA PARA SALAS SMT POR MEIO DA IOT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Orientador: Professor Esp. Jorge Luis Sarapka

Trabalho de Conclusão de Curso apresentado e aprovado em:_____/_____/2023

Banca Examinadora:

_____

Prof. Esp. Jorge Luis Sarapka, FATEC SBC – Orientador

_____

Prof. Esp. Marcos Vagner Zamboni, FATEC SBC - Avaliador

_____

Prof. Me. Rômulo Oliveira Albuquerque, FATEC SBC - Avaliador

Dedicamos esse trabalho aos nossos pais e esposas, e a todos os professores aqueles que me ajudaram e aos colegas que de alguma forma também ajudaram.

Agradeço ao prof. Especialista Jorge Luis Sarapka pela ajuda durante a elaboração deste trabalho.

“Há duas formas de viver a vida: uma é acreditar que não existem milagres, a outra é acreditar que todas as coisas são um milagre”

ALBERT EINSTEIN

RESUMO

A pesquisa tem como objetivo geral a elaboração de um protótipo que realize o controle e monitoramento de ambientes utilizando a ESP32, no qual pode ser aplicado na indústria de modo geral, porém com uma maior ênfase em salas com aplicação de solda em placas eletrônicas. Com o uso da eletrônica e da programação os sensores coletam informações a todo o instante da temperatura e umidade do ambiente, com os set-points definidos dentro do padrão de melhor temperatura e umidade, os atuadores mantém o ambiente dentro da faixa predeterminada, além da visualização e armazenamento dos dados no supervisório os dados também serão disponibilizados na rede utilizando os conceitos de IOT e computação em nuvem. É apresentado ao longo de toda a monografia a programação, o processo de comunicação, criação da interface no supervisório, criação da interface na nuvem e a criação da parte estrutural do projeto. Para a elaboração deste trabalho as pesquisas bibliográficas forneceram dados para dar sustentação na elaboração do trabalho, a metodologia cientifica foi essencial para a produção do protótipo, que se mostrou eficaz para a solução proposta.

Palavras-chave: Programação. SMT. Nuvem. ESP32. Eletrônica.

ABSTRACT

The general objective of the research is the elaboration of a prototype that performs the control and monitoring of environments using the ESP32, in which it can be applied in the industry in general, but with a greater emphasis on rooms where soldering is applied to electronic boards. With the use of electronics and programming, the sensors collect information at all times on the temperature and humidity of the environment, with the set-points defined within the best temperature and humidity standard, the actuators maintain the environment within the predetermined range, in addition to the visualization and storage of data in the supervisory the data will also be made available on the network using the concepts of IOT and cloud computing. The programming, the communication process, creation of the interface in the supervisory, creation of the interface in the cloud and the creation of the structural part of the project are presented throughout the monograph. For the elaboration of this work, the bibliographical researches provided data to support the elaboration of the work, the scientific methodology was essential for the production of the prototype, which proved to be effective for the proposed solution.

Keywords: Programming. SMT. Cloud. ESP32. Electronics.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

O controle e monitoramento em salas SMD (Dispositivos de Montagem Superficial) é de extrema importância devido as variáveis naturais capazes de afetar diretamente o processo, diante de tal situação pensamos em desenvolver uma alternativa de baixo custo de software e hardware livres, que possibilitam o monitoramento, controle e gerenciamento da temperatura ambiente para aplicação em salas de soldagem. O projeto visa assegurar uma climatização e umidade dentro de uma faixa específica, com temperaturas satisfatórias para o bom funcionamento dos equipamentos e detectar qualquer anormalidade das grandezas físicas medidas, por meio de notificação do monitoramento remoto e assim corrigir rapidamente.

As funções básicas de um sistema de automação em uma área limpa, são monitorar as condições do ambiente e controlar os demais dispositivos para mantê-los de acordo com o set point determinado. Para o controle, sensores serão posicionados estrategicamente e enviarão as informações de forma contínua e em tempo real para o supervisório, ao receber os sinais fora da faixa determinada os atuadores serão ligados para correção seja da temperatura ou umidade, para garantir que as condições do local permaneçam nos valores ideais, conforme a atividade que está sendo desenvolvida.

Considerando a importância de manter a sala dentro de condições ideais, a ideia será criar uma interface homem-máquina onde será feito o controle e o monitoramento da temperatura e umidade do local, assim será possível a garantia de produtos feitos em SMT (Tecnologia de Montagem Superficial) com melhor qualidade e aumentar a sua vida útil.

O IoT (Internet das coisas) atuará captando informações através dos sensores e enviando para a nuvem esses dados, assim tendo acesso a qualquer dispositivo. A

versatilidade desta solução via Wi-fi simplifica o processo de monitoramento e a partir da análise das informações, tem-se uma visão do processo além de auxiliar melhor nas tomadas de decisões. A importância de monitorar esses dados e armazenar nas nuvens seria para uma fácil visualização dos dados obtidos em tempo real, além do armazenamento de um histórico para coleta e tratativa dos dados.

O trabalho é dividido em três capítulos:

Capítulo 1 – Fundamentação teórica: faz-se uma abordagem das teorias que dão sustentação ao desenvolvimento do projeto.

Capítulo 2 – Metodologia: descreve o caminho a percorrer para a elaboração do projeto, destacando métodos, técnicas e procedimentos.

Capítulo 3 – Desenvolvimento do projeto: descreve passo a passo a construção e desenvolvimento do projeto, bem como figuras e tabelas para melhor entendimento.

E finalmente, as considerações finais: são descritos os objetivos propostos e atingidos, justificativa, pontos fortes e fracos, conquistas alcançadas relações entre as teorias e os fatos verificados e possíveis sugestões para futuros projetos.

1 FUNDAMENTAÇÃO TEORICA

Neste capítulo são abordadas teorias de autores que dão sustentação ao desenvolvimento e construção do projeto ambiente com temperatura e umidade controlada para salas SMD por meio da iot.

1.1 Montagem de SMD

De acordo com Mattede (2019), SMD (*Surface Mounted Device*- Dispositivo de Montagem em Superfície) que faz parte do processo (*SMT-Surface Mounting Tecnology* - Tecnologia de Montagem em Superfície). Essa inserção é feita com componentes ultraminiaturizados, que ao invés de serem fixados com furos eles são soldados diretamente na superficie de um lado da PCI (Placa de Circuito Integrado), o que economiza muito espaço e reduz bastante o tamanho dos circuitos eletrônicos, abaixo alguns exemplos de códigos e medidas na Figura 1.1.

Figura 1.1 – Componentes e a tabela de códigos e medidas

CÓDIGO	MEDIDA
EM POLEGADA	
0201	0,02" x 0,01"
0402	0,04" x 0,02"
0504	0,05" x 0,04"
0805	0,08" x 0,05"
1206	0,12" x 0,06"
1802	0,18" x 0,02"
2225	0,22" x 0,25"

Fonte: https://www.mundodaeletrica.com.br/

Os componentes são os mesmos que os convencionais, sendo usados nos mesmos circuitos eletrônicos como por exemplo, placas de tabletes, celulares, notebooks, computadores entre outros.

Algumas vantagens:

- Montagem simples;
- Eles permitem componentes de pequena dimensão;
- Para fixá-los não precisa furar a PCI (Placa de Circuito Integrado);
- Eles têm uma certa resistência à choques mecânicos e vibrações;
- Ocupam menos espaço do que os convencionais;
- É possível fazer a montagem automática desses componentes;
- Redução de custo de montagem e de fabricação dos componentes;
- É possível colocar componentes dos dois lados da PCI.

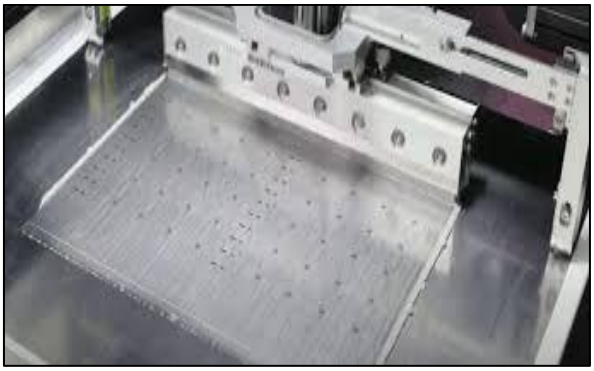
Algumas desvantagens:

- A manutenção e a troca em alguns casos são mais difíceis;
- O projeto do layout da PCI fica mais complexo;
- Os componentes não são bem padronizados;
- Eles têm uma certa dificuldade em fazer a dissipação térmica;
- A identificação por legenda é prejudicada, podendo até não ser feita.

1.1.1 Aplicação da pasta de solda

Gouveia (2018), a aplicação de pasta de solda é um dos processos mais importantes da montagem SMT, A pasta de solda é aplicada por uma printer que é um tipo de impressora. Durante o processo a máquina utiliza um rolo que realiza a aplicação com a ajuda de uma chapa de metal com aberturas que são na verdade o desenho do Pad’s (Pontos onde serão soldados os componentes eletrônicos) onde será colocada a pasta de solda, esta chapa é chamada de stencil e pode ser visualizada na Figura 1.2.

Figura 1.2– Máquina printer



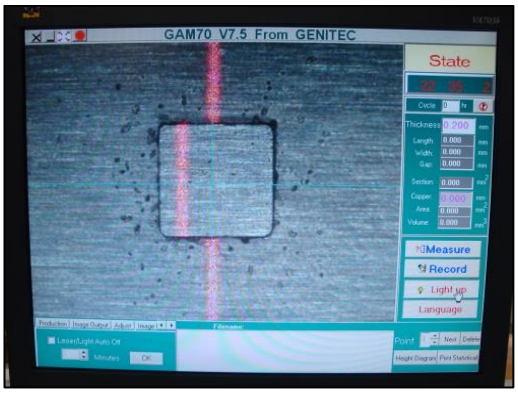
Fonte: www.softmontagem.com.br

Logo após esse processo o painel seguirá para o próximo passo onde o painel irá passar por uma inspeção da solda.

1.1.2 Inspeção da pasta de solda

Gouveia (2018), explica que os processos de inspeções são vantajosos do processo de fabricação SMT, porque evitam que a placa siga na linha de montagem com alguma inconsistência ou defeito. Sendo assim, fica fácil corrigir os erros pontualmente ao invés de ter que desmontar várias partes do produto ou mesmo descartar a peça.

Figura 1.3 – Tela de inspeção de solda



Fonte: allponi.com.br

Na Figura 1.3 pode-se observar a etapa de inspeção, é necessária a utilização de um SPI (*Solder Paste Inspection* - Inspeção da Pasta de Solda), um equipamento que permite, por meio da captura de imagens da placa, analisar a correta disposição da pasta de solda utilizando uma técnica de design gráfico que utiliza um molde vazado para criar imagens, essa técnica é conhecida como stencil. Assim, é possível determinar a quantia correta de pasta de solda aplicada, caso o resultado seja negativo, a placa será retirada de linha e fabricação em SMT. O processo será paralisado e a placa deverá passar por uma limpeza antes de iniciar novamente o primeiro processo de aplicação de solda.

1.1.3 Inserção de componentes

De acordo com Gouveia (2018), a fabricação em SMT conta com um equipamento especial para a etapa de inserção de componentes a Insersora, desta forma, os componentes são coletados e depositados de acordo com a sua correta posição na placa. Para isso, é realizada uma programação com coordenadas de acordo com o tamanho da placa. O equipamento ainda é capaz de reconhecer o deslocamento durante a coleta e promover a correção durante a montagem do componente como pode-se observar na Figura 1.4.

Figura 1.4 – Inserção de componentes SMD

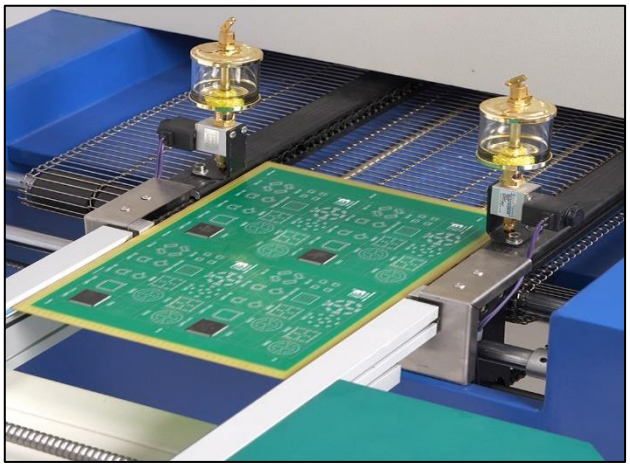
Fonte: www.vetron.com.br

Desta forma os componentes são colocados em suas respectivas posições sem a possibilidade de erros pois o programa segue todas as coordenadas de acordo com o algoritmo estabelecido.

1.1.4 Refusão

Para Gouveia (2018), a função do forno de refusão é a soldagem automática dos terminais dos componentes com os PAD's, O que ocorre é que até este momento a placa de circuito impresso já recebeu a pasta de solda nas regiões a serem soldadas, mas os terminais dos componentes não estão soldados, ou seja, eles estão apenas encostados na pasta de solda, na Figura 1.5 pode–se observar uma placa entrando no forno para começar o processo de refusão.

Figura 1.5 – Refusão de componentes SMD



Fonte: https://es.123rf.com,2022

A temperatura ideal para fundir a pasta de solda e realizar a conexão eletromecânica dos terminais dos componentes nos PAD's da PCI é regulada no forno de refusão, para garantir o sucesso da etapa, o forno passa por uma série de testes com "placas de sacrifício" até chegar às zonas de aquecimento corretas. A placa ainda é instrumentada com termopares em seus pontos críticos, pois, assim, será possível avaliar a temperatura e permitir que os componentes sejam soldados adequadamente e sem excesso de exposição.

1.1.5 Inspeção Optica Automatizada

Segundo Gouveia (2018), a fase final da fabricação em SMT é chamada de AOI (Automated Optical Inspection- Inspeção Optica Automatizada), é o momento em que ocorre a inspeção dos componentes, essa etapa consiste em capturar imagens da placa e dos componentes, a precisão e a repetibilidade são méritos inerentes à esse tipo de montagem sendo assim a AOI é a prova final de que tudo ocorreu como esperado e o equipamento não apresentará defeitos em relação ao componente eletroeletrônico quando chegar ao consumidor final.

Figura 1.6 – Máquina de inspeção automática

Fonte: www.pcbdirectory.com

Como mostrado na Figura 1.6 logo acima, a inspeção ocorre por meio de um programa desenvolvido com placas modelos realiza a comparação com o material produzido, ou seja, a placa produzida precisa ser exatamente igual àquela que foi colocada como modelo. Se houver diferenças, o operador receberá um aviso para corrigir o problema. Entre os erros detectados nesta etapa estão: componentes trocados (devido a diferença na serigrafia), componentes montados invertidos, falta de componentes, possíveis curtos-circuitos na solda, excesso de solda e falta de solda.

1.2 Controle de temperatura e umidade

De acordo com Gouveia (2018), a classificação MSL (*Moisture Sensitivity Level* - Nível de sensibilidade à umidade) é uma ferramenta criada pela indústria eletrônica para descrever quanto tempo um dispositivo sensível à umidade pode ser exposto a condições de temperatura ambiente (aproximadamente 30°C e 60% de umidade relativa). Seguir essa regra não é tarefa simples, pois traz consigo algumas restrições quanto à embalagem, manuseio e armazenagem dos dispositivos semicondutores.

Todo o material de plástico é capaz de absorver mais ou menos quantidade de umidade do ambiente, o que pode se tornar um problema para a montagem eficaz de placas. O excesso de umidade pode se transformar em vapor e criar defeitos, reduzindo o rendimento da produção. Controlar essa quantidade absorvida, portanto, impacta diretamente na qualidade final do produto.

Esses componentes absorvem a umidade do ar e podem ser danificados durante a soldagem da placa por refusão, o aumento da umidade em processamentos de alta temperatura pode resultar na formação de bolhas (popcorn effect), fissuras (separação interna) ou delaminações (no caso de placas nuas). Estes defeitos internos são difíceis de serem detectados durante o processo de montagem de placas e teste, o que gera um impacto negativo sobre a produção e montagem das placas eletrônicas, e posteriormente, sobre a venda dos produtos.

De acordo com Michell (2018), o controle de temperatura e umidade feito em um setor de montagem de componente SMD, o critério de temperatura fica entre 20C° no mínimo e 26C° no máximo, no caso da umidade o critério é manter uma umidade relativa entre 40% e 60%.

Esse mesmo controle também serve para manter o maquinário em funcionamento constante sem que haja maiores complicações, pois, as máquinas sofrem um aquecimento durante o processo e que se não tiver um controle poderá vir a causar defeito.

1.2.1 Temperatura inadequada

O amplo uso de placas de circuitos impressos para funcionamento de equipamento eletrônico, necessitam de cuidados especiais, principalmente quando instaladas em ambientes agressivos sujeitos a umidade e temperaturas críticas. Elevados gradientes de temperatura induzem falhas nos diversos arranjos eletrônicos associadas a tensões termicamente induzidas, assim tipificadas:

- Falha nas junções dos transistores;
- Deformações elásticas ou plásticas excessivas;
- Falha de ruptura dúctil e frágil;
- Falha de fadiga térmica;
- Falha devido a choque térmico;
- Falha devido a tensões de corrosão.

Em consequência disso, o controle de temperatura e os níveis de umidade relativa no ambiente é a chave para ter impacto significante em custo, confiabilidade, durabilidade (ESSS,2008).

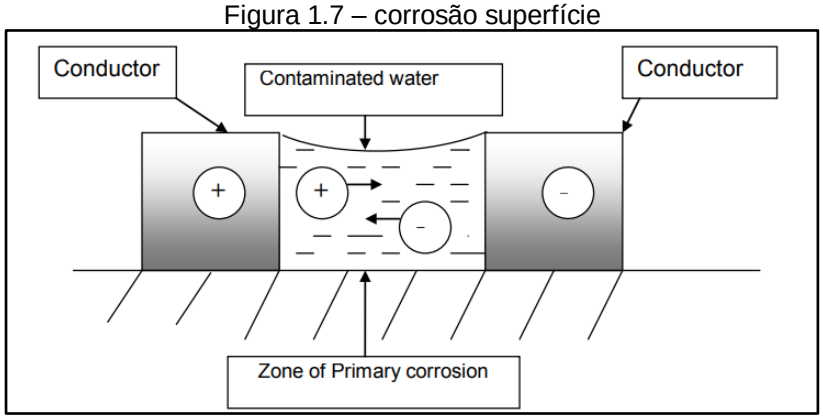
1.2.1 Absorção de umidade

Santos (2017), explica que a umidade é medida pela RH (Relative Humidity - Umidade Relativa) da sala, que é a relação da pressão parcial do vapor de água à pressão de vapor de equilíbrio da água à mesma temperatura. A faixa de umidade recomendada é determinada pelas especificações dos componentes e dispositivos que estão sendo montados, de acordo com Norma IPC/JEDEC J-STD-033B01.

O alto nível de umidade pode influenciar o desempenho dos eletrônicos, incentivando a corrosão, Figura 1.7, do envelhecimento na superfície do conjunto da

placa de circuito impresso. Pois eles atuam com cargas elétricas e essas cargas sofrem com a alteração da umidade do ambiente, tendo como consequência mais comum a descarga eletroestática (ESSS,2008).

Figura 1.7 – corrosão superfície



Fonte: VIMALA, NATESAN e RAJENDRAN, 2009

Segundo Michell (2018), em condições de umidade elevada, a pasta de solda absorve água do ambiente, resultando em integração deficiente, caindo da pasta, e a formação de esferas de solda durante a refusão. As temperaturas elevadas reduzem a viscosidade da pasta de solda, aumentando a incidência de manchas e quedas, articulações ultrapassadas e esferas de solda.

Portanto, para um melhor projeto de controle climático, um sistema de ar-condicionado para controle de umidade relativa e temperatura é necessário**.**

1.3 Sensores de temperatura e umidade de baixo custo

De acordo com Thomazini e Albuquerque (2012), um sistema industrial é preciso determinar as condições (ou variáveis) do sistema. É necessário obter os valores das variáveis físicas do ambiente a ser monitorado, e este é trabalho dos sensores.

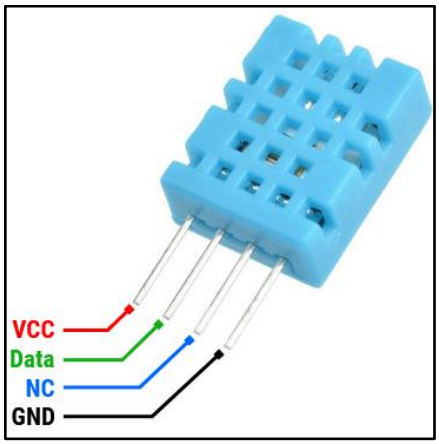
Os sensores medem uma grandeza física e entregam um sinal elétrico como saída. Se esse sinal puder tomar qualquer valor dentro de certos limites ao longo do tempo, esse sensor é chamado de analógico. Caso esse sinal elétrico puder somente tomar dois valores ao longo do tempo, sejam estes sinais de qualquer amplitude, o sensor é chamado de digital.

Em seguida, terá dois modelos de sensores de baixo custo para monitoramento de temperatura e umidade relativa do ar que poderá ser utilizado nesse projeto.

1.3.1 Sensor DTH11

O sensor DHT11 é um sensor de dupla ação, ou seja, ele consegue detectar duas variações ao mesmo tempo como a temperatura e umidade, fabricados pela Aosong Electronics Co. Ltd. Este sensor se destaca por ser componente que possui um nível de estabilidade muito bom e recursos de calibração muito precisos, rápida resposta além de um baixo custo, conforma ilustra a figura 1.8.

Figura 1.8 - Sensor DHT11 pinos



Fonte: www.circuitgeeks.com

Aosong (2016), explica que o sensor tem seu funcionamento ativado com 3 a 5,5 VDC (Volts em Corrente Contínua), sendo recomendado pelo fabricante o uso dele em 5,5 VDC, tem seu range de trabalho de temperatura entre 0 a 50ºC erro de ± 2 °C e de 20-90% ± 5% erro para a umidade relativa do ar (UR).

O DHT11 contém 4 terminais, mas no seu funcionamento é utilizado apenas 3 pinos, que são: o pino de alimentação (Vcc), pino terra (GND - ground) pino de envio. Alguns fabricantes recomendam o uso de um resistor de pelo menos 5kohms (kΩ), no pino do sinal ao fazer a conexão do sensor ao microcontrolador.

1.3.2 Sensor DTH22

E assim como o DHT11 o DHT22 (figura 1.9) é um sensor de Temperatura e umidade fabricados pela Aosong Electronics Co. Ltd. É reivindicado como tendo qualidade de leitura bom, rápida resposta (2s), é calibrado de fábrica e não necessita de hardware adicional para funcionar, e além de um preço relativamente barato.

Figura 1.9 - Sensor DHT22 pinos

Fonte: www.circuitgeeks.com

Aosong (2016), explica que o sensor tem seu funcionamento ativado com 3 a 6 VDC (Volts em Corrente Contínua), tem seu range de trabalho de temperatura entre entre - 40 a 80ºC erro de ± 0,2 °C e de 0-100% ± 2% erro para a umidade relativa do ar.

O DHT22 contém 4 terminais, mas no seu funcionamento é utilizado apenas 3 pinos, que são: o pino de alimentação (Vcc), pino de envio e o pino terra (GND -

ground). Alguns fabricantes recomendam o uso de um resistor de pelo menos 10kohms (kΩ), no pino do sinal ao fazer a conexão do sensor ao microcontrolador, para que se tenha um melhor efeito de manutenção da calibração.

1.3.3 Comparação entre os sensores

De acordo com Autocore (2017), conforme analisar a Figura 1.10, os sensores podem ser alimentados com tensões 3.3V e 5V, o DHT22 possui faixas de leitura de temperatura e umidade maiores em relação ao DHT11, além de ter uma melhor precisão de leitura, possui também um consumo de corrente menor, já em contrapartida o DHT11 é mais rápido em ralação ao DHT22.

Figura 1.10 – Comparação dos sensores

	DHT11	DHT22
		
Tensão de Alimentação	3V - 5.5V	3.3V - 6V
Corrente Máxima(Captura Dados)	2.5mA	1.5mA
Faixa de Leitura - Umidade	20 - 80%	0 - 100%
Precisão - Umidade	5%	2%
Faixa de Leitura - Temperatura	0 - 50°C	- 40°C - 125°C
Precisão - Temperatura	+/- 2°C	+/- 0.5°C
Intervalo entre Medições	1s	2s

Fonte: https://autocorerobotica.blog.br/

Para nosso projeto de controle de temperatura e umidade feito em um setor de montagem de componente SMD, onde o range de temperatura é de 20C° no mínimo e 26C° no máximo e umidade é de 40% no mínimo e 60% no máximo, pode-se optar pelo sensor DTH11 pois além de atender as necessidades possui um custo inferior.

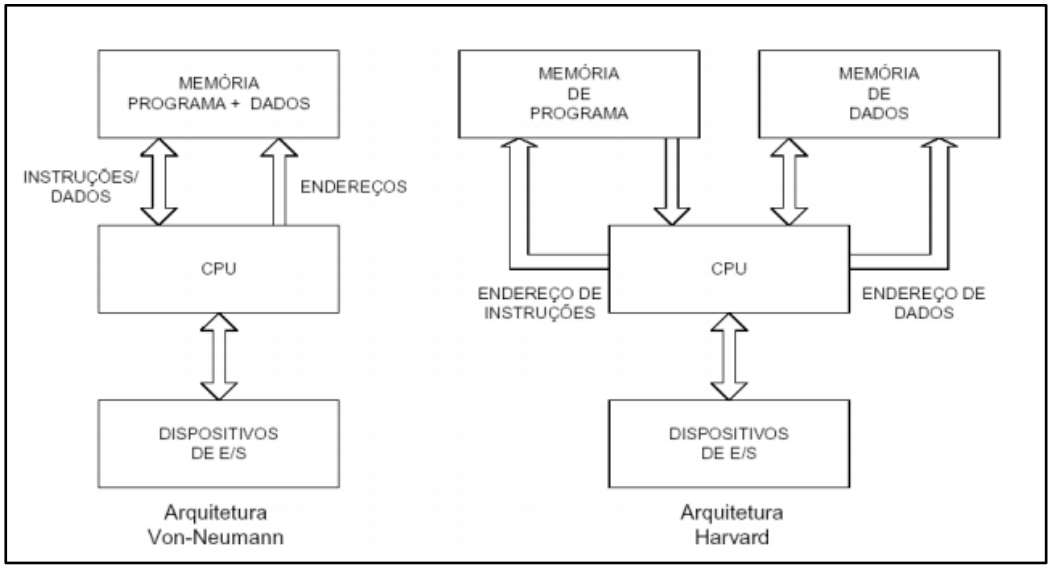
1.4 Microcontrolador

Kerschbaumer et al. (2017), relata que o microcontrolador é um pequeno computador que é composto basicamente por uma unidade central de processamento, memórias, e o circuito de clock, entradas e saídas (I/O), conversores analógicos, digitais e que tem como função realizar ações de controle de maneira remota em sistemas embarcados, basicamente é um pequeno computador embutido e que tem baixo custo.

Quando se fala em arquitetura, existem dois tipos em destaque, a Von Neuman e a arquitetura Harvard, basicamente a diferencia, é a forma como a memória é conectada ao microprocessador.

Na arquitetura de Von Neuman, Figura 1.11 – lado esquerdo, onde a memória é unificada, são armazenadas tanto as instruções quanto os dados, ou seja, para leitura e escrita de dados, por outro lado, a arquitetura de Harvard, Figura 1.11 – lado direito, que tem a separação da memorias com armazenamento de dados que garante maior velocidade de processamento.

Figura 1.11 - Arquitetura Von Neuman e Harvard

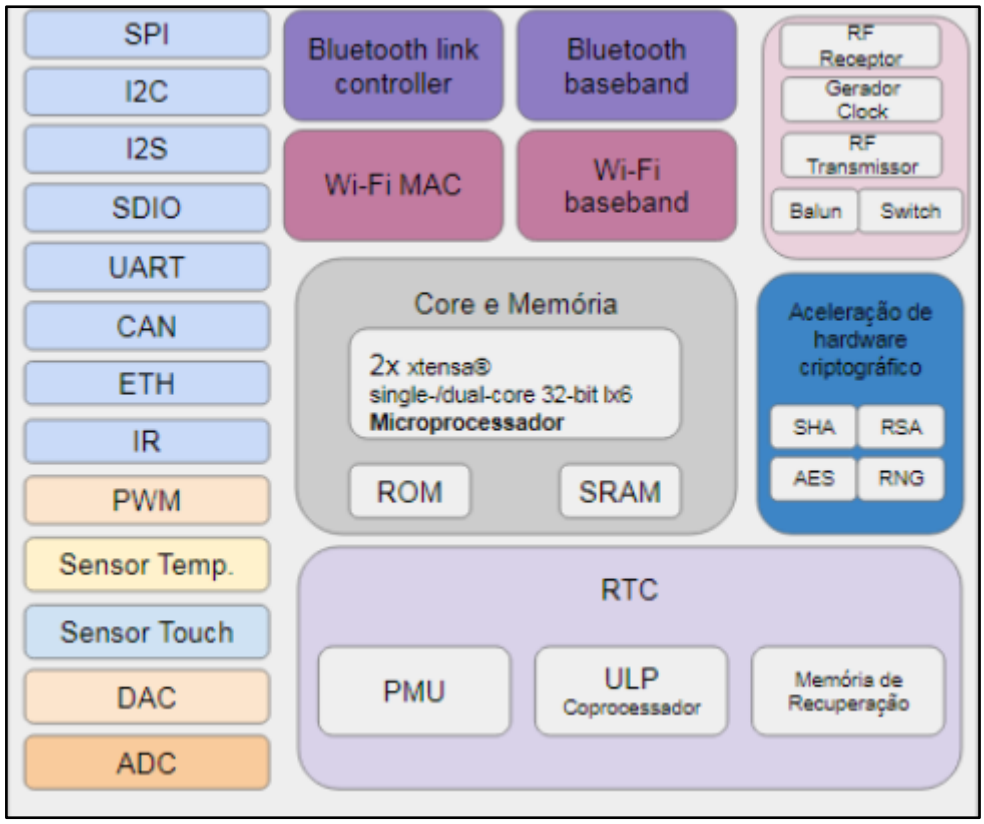


Fonte: https://www.feis.unesp.br/

Desenvolvido pela empresa Espressif (2018), o ESP 32 é um microcontrolador, versátil e de alta performance pois tem a capacidade de comunicação sem fio através do WI-FI e bluetooth, e que conta com dois processadores Xtensa LX6 (o popular dual-core) com arquitetura (32 bits) de Harvard.

Possui clock interno de até 240 MHz, tendo Memória RAM de 512 KB, memória Flash de 16Mb, tem 48 pinos em sua arquitetura além do terra, nove interfaces de saída PWM, dezoito entradas ADC (analógico para digital), além de duas saídas DAC (digital para analógico), conforma ilustra a Figura 1.12.

Figura 1.12 - Arquitetura ESP 32



Fonte: https://curtocircuito.com.br/

A programação do ESP32 é pode ser feita através da linguagem de programação C/C++, que pode ser desenvolvida através do software Arduíno IDE, com uma estruturada conectividade sem fio é possível realizar a programação remota, utilizando por exemplo o wi-fi.

Tomando como referência os fatos mencionados a viabilidade da utilização do ESP32 como microcontrolador para a execução de um projeto de baixo custo e alta conectividade com os componentes do sistema de automatização e tornando a construção de sistema como internet das coisas (IoT) muito mais simples e compacto.

1.5 Internet das coisas

Sendo a Techtudo (2022), IOT (*Internet of Things* – Internet das coisas) é um termo usado para definir a tecnologia que conecta itens do dia a dia à internet, como por exemplo celulares, eletrodomésticos, meios de transportes, televisores entre outros, assim integrando o mundo físico ao digital.

Essa ideia surgiu em 1991 com a internet e a conexão TCP/IP (*Transmission Control Protocol/Internet Protocol* – Protocolo de Controle de transmissão/Protocolo da Internet) se tornando mais populares, Bill Joy, cofundador da Sun Microsystems pensou sobre a conexão (D2D) Device para Device tipo de ligação que faz parte de um conceito maior, o de “várias webs”.

O termo Internet das coisas foi criado por Kevin Ashton em 1999, Ashton dizia que em virtude da rotina e a limitação do tempo as pessoas iriam se conectar á internet de outras maneiras.

Um bom exemplo de aplicação da internet das coisas seria o protótipo Mobii que foi desenvolvido em 2014 através de uma parceria entre Ford e Intel que visava reinventar o interior de seus automóveis ,com essa tecnologia , o motorista ao entrar no veículo passaria por um reconhecimento facial através de uma câmera, caso o motorista seja reconhecido o sistema lhe oferece informações sobre seu cotidiano, recomendar músicas e receber orientações para acionar o mapa com GPS , se o motorista não for reconhecido o sistema tira uma foto e manda as informações para o celular do dono, evitando furtos.

A internet da coisa ainda pode ser usada em muitas outras áreas como agricultura fazendo o monitoramento de temperatura e umidade do solo, também pode ser usada na área da saúde fazendo integração de prontuários dos pacientes e relatórios dos médicos.

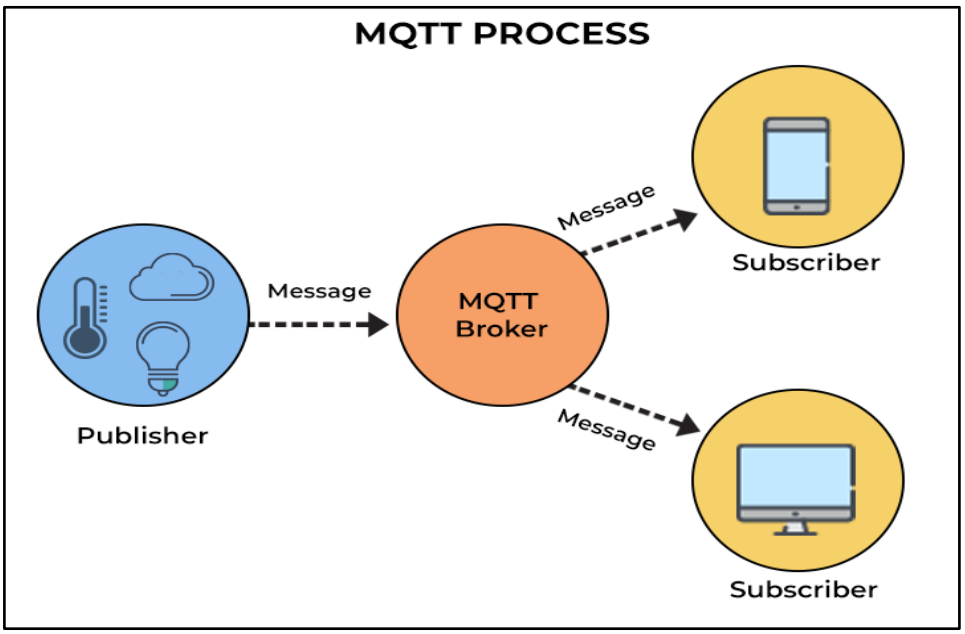
1.5.1 Protocolo MQTT

OASIS (2014), na especificação oficial do MQTT disponível no site mqtt.org, protocolo MQTT (Message Queuing Telemetry Transport) é um protocolo de mensagens leve e de publicação/assinatura projetado para comunicações entre dispositivos de Internet das Coisas (IoT). Ele foi desenvolvido pela IBM e agora é um padrão aberto mantido pela OASIS (Organization for the Advancement of Structured Information Standards).

O MQTT opera em cima de protocolos de transporte TCP/IP e é projetado para ser eficiente em termos de consumo de energia, largura de banda e requisitos de processamento. Ele é especialmente adequado para cenários em que os dispositivos de IoT possuem recursos limitados, como sensores, medidores inteligentes e dispositivos de monitoramento.

O modelo de comunicação no MQTT é baseado em um sistema de publicação/assinatura. Os dispositivos que desejam enviar mensagens (publicadores) enviam-nas para um intermediário central chamado broker MQTT. Essas mensagens são publicadas em um tópico específico. Por outro lado, os dispositivos que desejam receber as mensagens (assinantes) se inscrevem nesses tópicos específicos. O broker encaminha as mensagens publicadas para todos os assinantes inscritos nesses tópicos, conforma ilustra a Figura 1.13.

Figura 1.13 - sistema de publicação/assinatura



Fonte: https://www.spiceworks.com/tech/iot/articles/what-is-mqtt/

BasuMallick (2022), O MQTT suporta três níveis de qualidade de serviço (QoS) para garantir a entrega confiável das mensagens:

- QoS 0 (no máximo uma vez): As mensagens são enviadas uma única vez e não há garantia de entrega ou confirmação;

- QoS 1 (pelo menos uma vez): As mensagens são enviadas pelo menos uma vez e a entrega é confirmada pelo receptor. Se a confirmação não for recebida, a mensagem é reenviada;

- QoS 2 (exatamente uma vez): As mensagens são entregues exatamente uma vez, garantindo que não haja duplicação. Há uma troca completa de mensagens de confirmação entre o publicador e o assinante para garantir a entrega.

O MQTT também suporta recursos como retenção de mensagens, onde o broker armazena a mensagem mais recente para um tópico e a envia para novos assinantes assim que se conectam.

O protocolo MQTT possui uma estrutura de pacotes simples e eficiente. Cada pacote possui um cabeçalho fixo com informações sobre o tipo de pacote, flags e tamanho, seguido pelos dados da mensagem propriamente dita.

Existem bibliotecas MQTT disponíveis para várias linguagens de programação, permitindo que os desenvolvedores implementem o protocolo em diferentes dispositivos e plataformas.

1.6 Controle PID

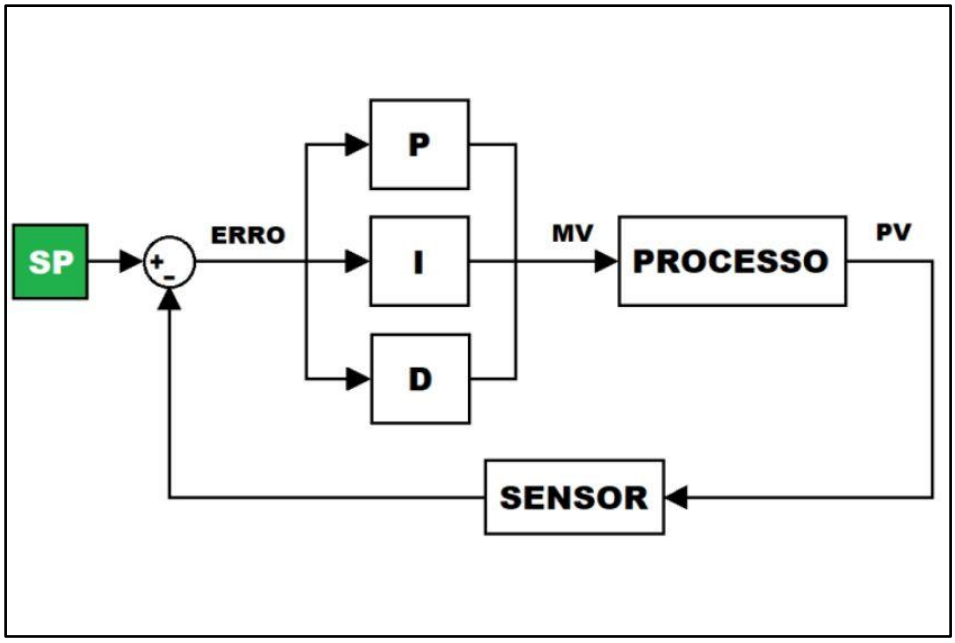
Damasio (2020), O controle PID (Proporcional - Integral - Derivativo) é uma técnica de controle amplamente utilizada na indústria para garantir a estabilidade e eficiência dos processos. Foi desenvolvido inicialmente por Elmer Sperry em 1911 para automatizar a direção dos navios da Marinha dos Estados Unidos. O controle PID combina três ações: proporcional, integral e derivativa, para gerar um sinal de controle que ajusta a saída do processo.

A ação proporcional faz com que o sistema reaja imediatamente ao erro presente, permitindo uma resposta rápida a variações e perturbações. A ação integral elimina os erros em regime permanente, garantindo que o sistema atinja o valor desejado a longo prazo. Já a ação derivativa antecipa o comportamento do processo, aumentando a estabilidade e tornando a resposta mais rápida.

Essas três ações são combinadas em uma equação matemática que calcula o sinal de controle. Cada ação utiliza o valor do erro para gerar o sinal. O controle PID

compensa as características indesejáveis de uma ação com as outras, resultando em um controle eficiente e estável, conforma ilustra a Figura 1.13.

Figura 1.14 – controle PID



Fonte: https://www.novus.com.br/blog/artigo-controle-pid-rompendo-a-barreira-do-tempo/

A combinação desses três termos, proporcional, integral e derivativo, forma o controle PID. A saída final é calculada como a soma ponderada dos três termos:

- Saída = Kp * (P + Ki * I + Kd * D);

- Onde Kp, Ki e Kd são os ganhos proporcional, integral e derivativo, respectivamente.

O controle PID é amplamente utilizado em diversas áreas, como automação industrial, robótica, engenharia de processos, controle de temperatura, entre outras. Ele oferece um método eficiente para ajustar automaticamente os parâmetros de controle e melhorar a resposta do sistema em relação a variações e perturbações.

2 METODOLOGIA

Neste capítulo encontra-se a trajetória para o desenvolvimento e construção do projeto que se intitula Ambiente com Temperatura e Umidade Controlada para salas SMT por meio da IOT. Trata-se de uma pesquisa aplicada que é desenvolvida nas dependências da FATEC São Bernardo do Campo e nas residências dos integrantes do grupo.

2.1 O que é Metodologia?

Prodanov e Freitas (2013) destacam que a metodologia é o caminho a percorrer para o desenvolvimento de uma pesquisa. Enfocam que os métodos são procedimentos amplos do raciocínio e as técnicas são procedimentos que operacionalizam os métodos mediante instrumentos adequados.

Severino (2017) enfatiza que a preparação metódica e planejada de um trabalho científico supõe uma sequência de etapas que compreende: determinação do tema-problema e justificativa, levantamento da bibliografia referente ao tema, leitura e documentação dessa bibliografia após seleção, construção lógica do trabalho e redação do texto.

A construção da redação do TCC tem como base o Manual de Normalização de Projeto de Trabalho de Graduação da FATEC - SBC (2022) que se encontra embasado nas normas da ABNT.

2.2 O tema-problema com justificativa e descrição do projeto

Embora a tecnologia esteja bem avançada para monitoramento de ambientes há ainda alguns setores onde esse monitoramento é feito de forma manual como por exemplo em salas de montagem de placas eletrônicas com componentes SMD, nesse tipo de sala a temperatura e a umidade precisam ser controladas pare que o maquinário não esquente demais e venha a dar defeito , normalmente os funcionários fazem esse controle manualmente através de um ar condicionado onde ele controla não só a temperatura como também a umidade , o processo começa no início do turno onde precisa-se manter a temperatura entre 20°C e 25°C e a umidade entre 40% e 60%, porem leva algum tempo até atingir essa temperatura e só depois disso liga-se o maquinário para começar o processo de produção , e também durante o dia deve-se monitorar esses parâmetros e fazer ajustes para mantê-los corretos.

Com o sistema de controle e monitoramento automatizado pode-se eliminar a necessidade de estar sempre ajustando esses parâmetro pois com ele isso será feito de forma automática, onde conforme a temperatura e a umidade do ambiente for mudando a sistema através se sensores e atuadores ira ajustar o clima para manter dentro dos parâmetros ,e todo esse monitoramento poderá ser acompanhado na tela de um supervisório através do computador ou smartfone facilitando serviço dos operadores e fazendo com que eles ganhem mais tempo no seu dia a dia .

Desta forma, principal justificativa para este trabalho se baseia no ganho competitivo no processo de monitoramento e a integração do conceito IOT.

O processo inicia-se com o microcontrolador ESP 32 e o sensor DHT11 coletando as informações de temperatura e umidade relativa do ar do ambiente a ser analisado. Os dados obtidos são enviados para o banco de dados via WiFi onde ficam salvas as informações, bem como horário e data da amostragem.

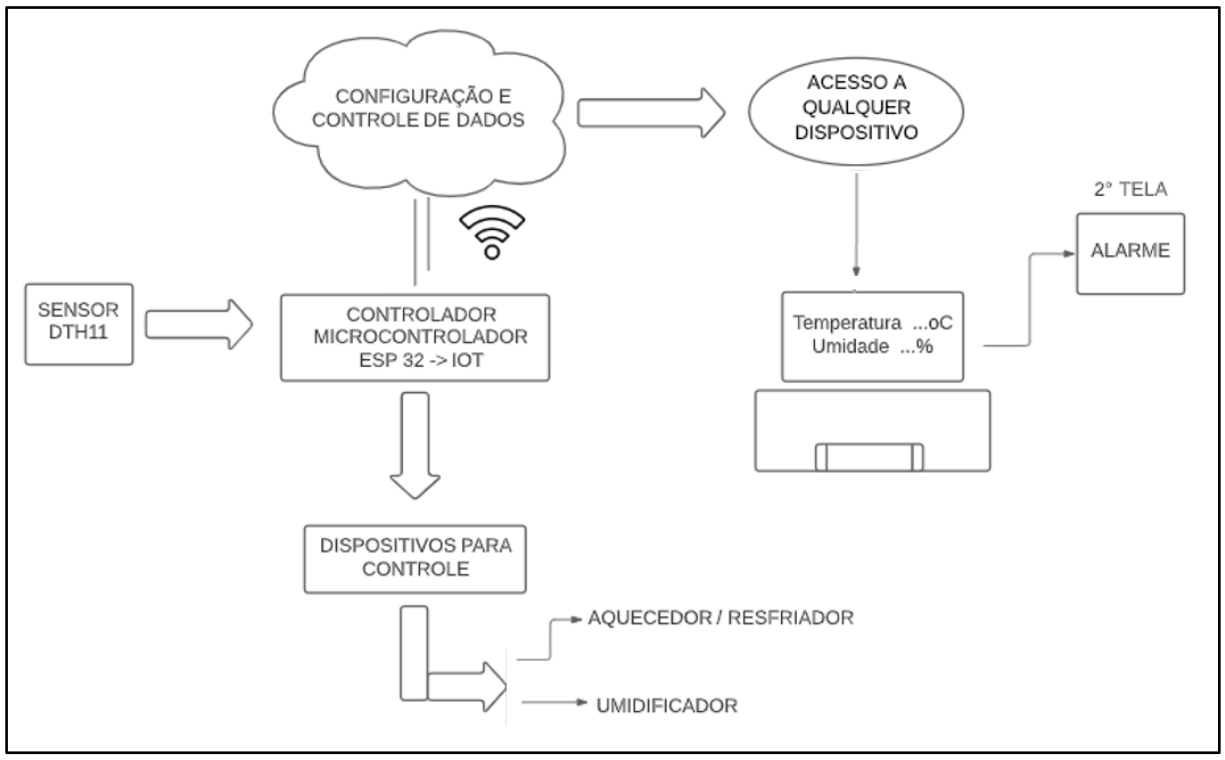
Através dos dados recebidos o sistema terá uma tela de alarme no supervisório indicando possíveis problemas de sobreaquecimento ou alta umidade no ambiente

indicando hora e data do ocorrido. Ao indicar esse alarme, automaticamente acionara os atuadores para o controle e assim regulando a temperatura e umidade.

Implementando a comunicação IOT junto ao projeto, faz-se com que os dados possam ser acessados remotamente por qualquer dispositivo.

Para a melhor visualização do projeto diagrama de blocos mostra a sua estrutura, conforme ilustra a Figura 2.1.

Figura 2.1 – Diagrama de blocos do projeto



Fonte: Autoria própria, 2022

Conforme o diagrama de blocos, o microcontrolador ESP 32 é o foco principal do diagrama, conectando a todos os blocos que funcionam da seguinte forma:

Controlador microcontrolador com IOT: o tipo de comunicação é feito via WiFi. O controlador é responsável pelo envio e recepção dos dados da nuvem para o controle e supervisão, receber os dados do sensor DTH11, e controlar o acionamento do aquecedor/resfriador e desumificador;

Configuração, controle e operação: é parte de IOT do projeto, ficando na “nuvem”;

O supervisório e acesso a outro dispositivo: o armazenamento de dados e o controle podem ser acessados remotamente, por meio da internet;

DTH11: É o sensor que vai monitorar set point desejado e, ao mesmo tempo, monitorado pelo controlador;

Dispositivos para controle: um tipo de Aquecedor/Resfriador e Desumificador que é controlado, estando com relação direto com o DTH11 e servindo no projeto para simular um sistema de Ar-condicionado.

2.2 Etapas teóricas e práticas para o desenvolvimento do projeto

Após o esclarecimento do tema-problema e uma breve descrição do projeto através do diagrama parte-se para as seguintes etapas:

Primeira etapa: reunião dos componentes do grupo, para escolha do orientador. Após o convite, o orientador se colocou à disposição do grupo e concordou nos orientar com o tema;

Segunda etapa: reunião do grupo com o orientador, para tratar de temas como métodos de pesquisa e possíveis referencias, além de uma explicação geral sobre o tema principal do projeto. O orientador colocou-se à disposição para suprimir algumas dúvidas e marcou, obrigatoriamente, um dia por semana para lhe apresentar o andamento da pesquisa;

Terceira etapa: o levantamento bibliográfico se deu na biblioteca da Fatec São Bernardo do Campo, em sites especializados, manuais e catálogos de empresas.

Quarta etapa: após leitura e releitura das bibliografias consultadas, fez-se uma seleção delas para construir o capítulo 1 –Fundamentação Teórica e Referências.

Quinta etapa: levantamento dos materiais e componentes que são utilizados na construção do projeto. Pesquisa em sites e lojas especializadas, buscando a melhor viabilidade de preço.

Tabela 1.1 – Preço componentes

Produto	Quantidade	Valor em Reais
Microcontrolador ESP32	1	48,99
Sensor DHT22	1	39,3
Lâmpada incandescente	1	10,59
Modulo de Alimentação	1	16,9
Mini kit Umidificador	1	25
Cooler	3	63,35
Placa Padrão 10x10	1	17,1
Chave seletora	1	2,30
Caixa Acrílico	1	150
Modulo dimmer	3	65.90
Cabo fonte	1	26.95
TOTAL		466,38

Autoria própria, 2023

Sexta etapa: configuração da plataforma do Arduíno para utilizar o ESP32.

Sétima etapa: Fazer Diagrama elétrico;

Oitava etapa: Download das bibliotecas para utilizar a programação do Microcontrolador e configuração da comunicação entre Microcontrolador (ESP32) e sistema IOT;

Nona etapa: Construção do protótipo para teste do sistema de monitoramento de temperatura e umidade;

Décima etapa: Testes e integração de todas as partes da programação;

Décima primeira etapa: Construção do protótipo físico para simulação da sala SMD;

Décima segunda etapa: Integração do protótipo físico e de toda a parte eletrônica.

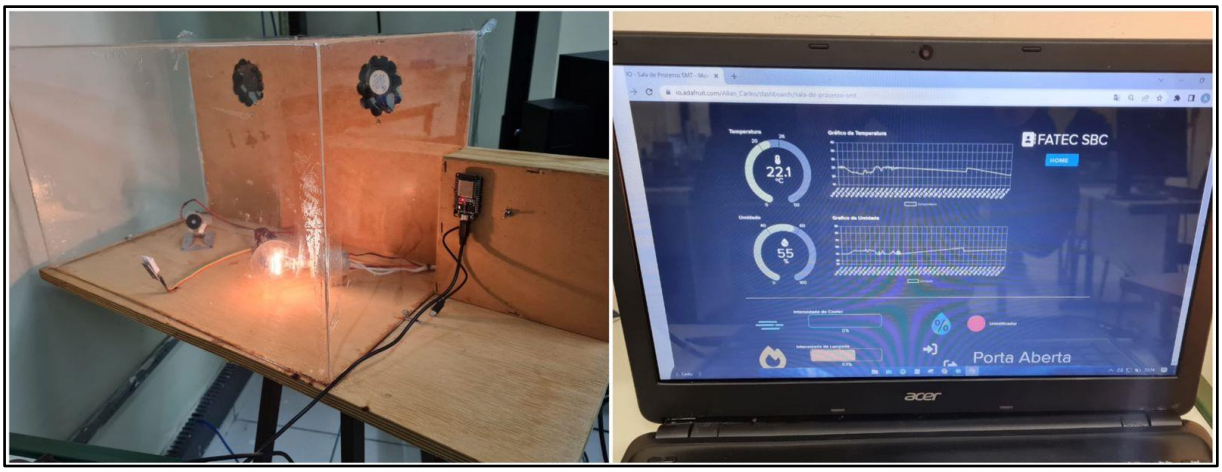
3 CONSTRUÇÃO DO PROTÓTIPO

Neste capítulo encontra- se passo a passo o desenvolvimento e construção lógica do projeto intitulado Ambiente com Temperatura e Umidade Controlada para salas SMT por meio da IOT.

3.1 O projeto finalizado

Para melhor entendimento a Figura 3.1 ilustra-o finalizado.

Figura 3.1 – O projeto finalizado



Fonte: Autoria própria, 2023

Seu funcionamento ocorre da seguinte maneira: o algoritmo foi todo realizado no microcontrolador ESP32, que tem a função de gerenciar todos os dados coletados e realizar os acionamentos.

No protótipo elaboradora há sensores, atuadores e o microcontrolador. A lâmpada faz a simulação de aquecimento do ambiente, enquanto os coolers e o umidificador atuam quando necessário, para controlar a temperatura e umidade do ambiente, enquanto o sensor da DHT11 realiza as medições em tempo real do ambiente.

O desenvolvimento do projeto encontra-se fundamentado nos seguintes tópicos:

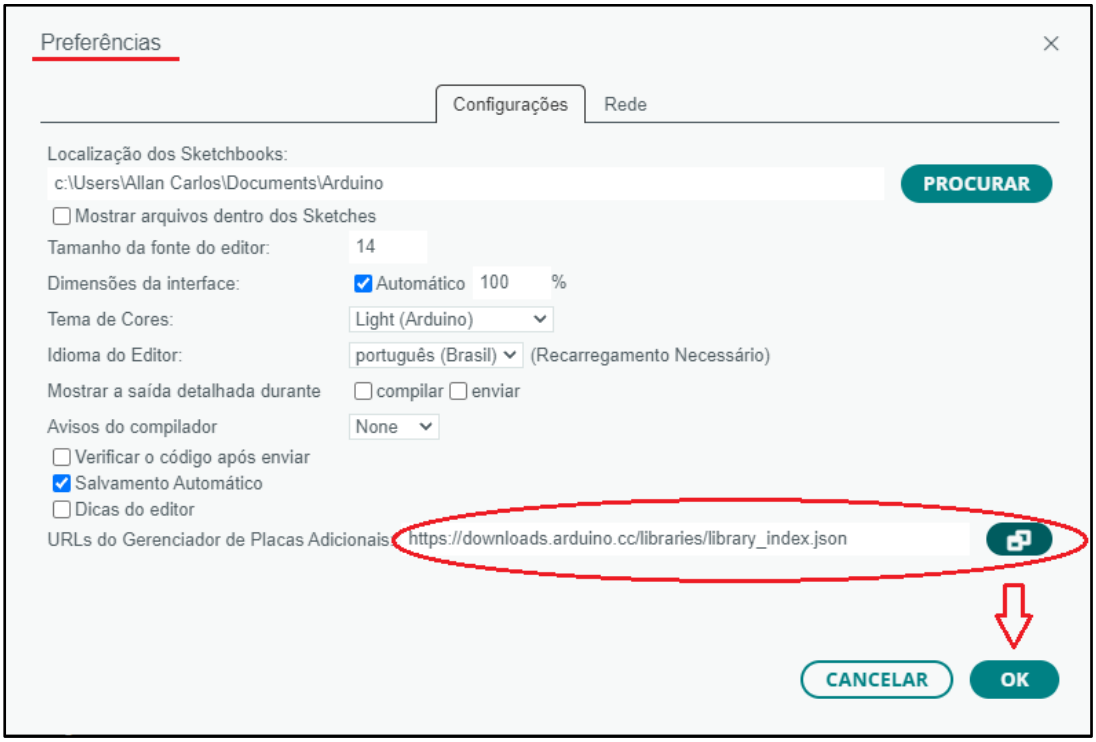
- Configuração da interface de programação do Microcontrolador
- Configuração microcontrolador e sistema IOT
- Construção do protótipo para teste utilizando ESP32 e DHT11
- Construção da estrutura do protótipo

3.2 Configuração da interface de programação do Microcontrolador

A primeira etapa do desenvolvimento refere-se à configuração da interface de programação do Microcontrolador ESP32, com intuito de utilizar o Arduino IDE. Faz-se uso de um conversor USB-serial para fazer a comunicação entre o Microcontrolador e o computador. Em seguida baixa-se o framework mais recente do Arduino, compatível com o computador que se utiliza. Realiza a instalação do framework do Arduino, abrindo o programa para iniciar a configuração.

Com a janela do Arduino IDE aberta, clica-se na aba “arquivo”, e logo após em “preferências”, um novo menu é aberto. Na parte inferior há uma caixa de texto descrita como “URLs do gerenciamento de placas adicionais”. Em seguida adiciona o endereço “https://downloads.arduino.cc/libraries/library_index.json”. Este endereço é disponibilizado pela Espressif para que o Microcontrolador seja utilizado na plataforma do Arduino. Clicando em “ok” fecha-se a tela e retorna-se ao início, conforme ilustra a Figura 3.1.

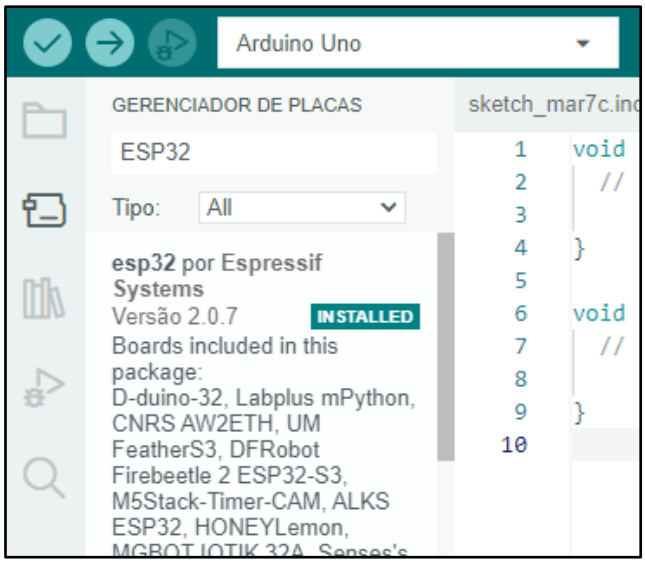
Figura 3.2 – ESP32 funcionando com o Arduino IDE



Fonte: Autoria própria, 2023

Na tela principal clica-se gerenciador de placas para instalar a biblioteca ESP32, em seguida digita-se “ESP32”. Clicando em instalar a biblioteca ESP32 está disponível para utilização, conforme ilustra a Figura 3.2:

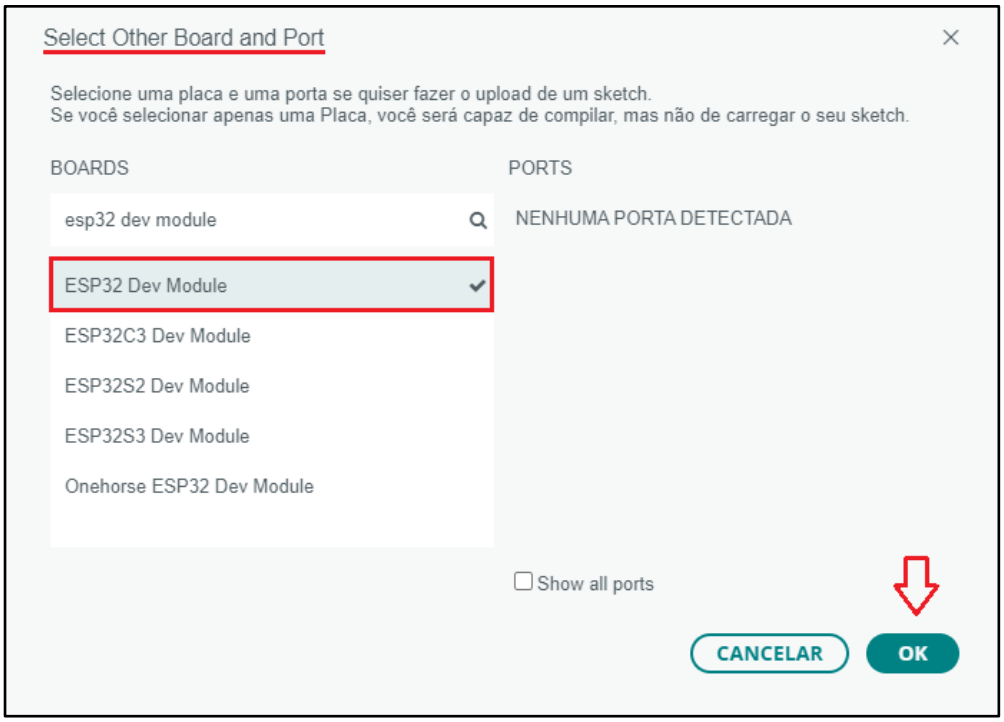
Figura 3.3 – Instalação ESP32



Fonte: Autoria própria, 2023

Em seguida seleciona-se a aba “selecione outra placa e porta” e um novo menu é aberto. Em seguida digita-se “ESP32” Uma lista é apresentada, seleciona-se “ESP32 Dev Module” e clica em “ok”, assim configurado faz-se a programação do Microcontrolador, conforme ilustra a Figura 3.4:

Figura 3.4 – ESP32 funcionando com o Arduino IDE



Fonte: Autoria própria, 2023

3.3 Configuração microcontrolador e sistema IOT

Após a configuração inicial do ESP32, o próximo passo é configurar a comunicação entre o microcontrolador e o sistema IoT escolhido. No caso deste projeto, utilizaremos o sistema IoT Adafruit IO. A primeira etapa para configurar a comunicação é criar uma conta de usuário na plataforma, isso pode ser feito acessando o site “https://accounts.adafruit.com/user/sign_up/” através de um computador e preenchendo o formulário de registro. Após o registro, é possível fazer o download da biblioteca “Adafruit_MQTT.h”, que será usada para enviar dados do microcontrolador para a plataforma Adafruit IO.

É importante manter o site da plataforma Adafruit IO aberto, pois serão necessárias informações específicas da sua conta, como o nome de usuário e chave de acesso, para configurar a comunicação entre o microcontrolador e a plataforma. Essas informações serão usadas para estabelecer a conexão e enviar dados para a plataforma.

Com a conta de usuário criada e a biblioteca baixada, é possível prosseguir com a configuração da comunicação entre o ESP32 e a plataforma Adafruit IO. Para isso, será necessário utilizar um código de exemplo fornecido pela biblioteca Adafruit_MQTT.h, que deve ser adaptado para se adequar às configurações específicas do projeto, conforme ilustra a Figura 3.5:

Figura 3.5 – Exemplo de uso da biblioteca do Adafruit_MQTT.h

```
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "...your SSID..."
#define WLAN_PASS       "...your password..."

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "...your AIO username (see https://accounts.adafruit.com)..."
#define AIO_KEY         "...your AIO key..."

/************ Global State (you don't need to change this!) ******************/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiClientSecure for SSL
//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/

// Setup a feed called 'photocell' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish photocell = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/photocell");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/onoff");
```

Fonte: Autoria própria, 2023

O último passo da configuração é definir a rede Wi-Fi para que o microcontrolador ESP32 possa se conectar à nuvem e enviar os dados do sensor para a plataforma Adafruit IO. Para isso, é necessário digitar o nome da rede Wi-Fi desejada no comando “#define SSID” do código, seguido pela sua respectiva senha de acesso no comando “#define SSID_PASSWORD”. É essencial inserir essas

informações corretamente para que a conexão Wi-Fi possa ser estabelecida com sucesso.

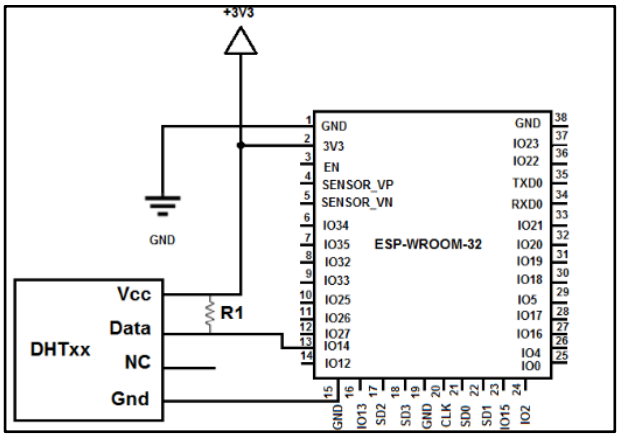
Uma vez que a comunicação entre o ESP32 e a plataforma Adafruit IO esteja configurada, o microcontrolador poderá enviar dados do sensor DHTxx para a plataforma, que poderá ser visualizado em tempo real em um dashboard personalizado.

3.4 Construção do protótipo para teste utilizando ESP32 e DHT11

Após configurar a IDE do Arduino e realizar as devidas configurações do ESP32 e da plataforma Adafruit IO, é possível dar início à construção do protótipo. O primeiro passo é criar o diagrama elétrico com o objetivo de construir fisicamente o protótipo através da protoboard.

Para isso, é preciso identificar os pinos do ESP32 e do sensor DHTxx que serão utilizados. O pino VCC do sensor DHTxx deve ser conectado ao pino 3.3V do ESP32, o pino DATA (ou sinal) do sensor DHTxx deve ser conectado ao pino D4 (GPIO4) do ESP32 e o pino GND do sensor DHTxx deve ser conectado ao pino GND do ESP32. Assim como mostra a figura 3.6.

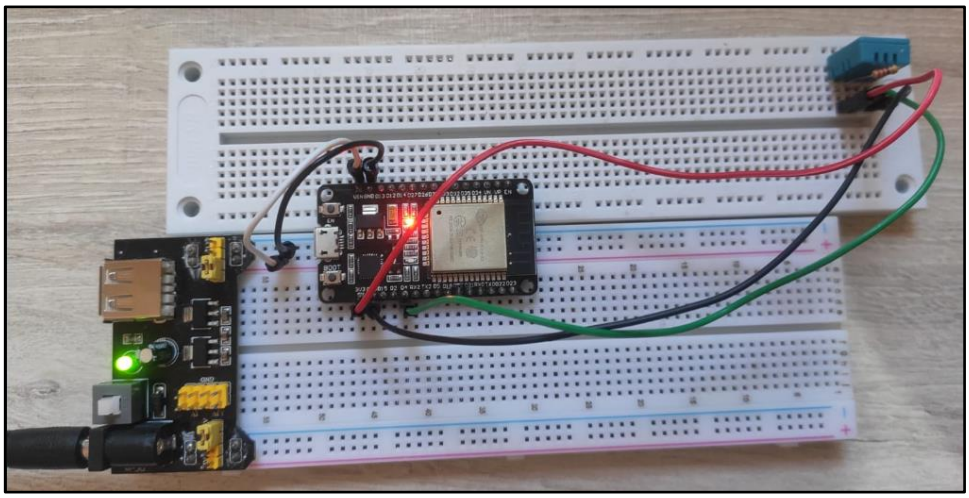
Figura 3.6 – Diagrama elétrico



Fonte: Autoria própria, 2023

Utiliza-se o diagrama elétrico da Figura 3.5 e uma protoboard para realizar a montagem eletrônica do protótipo, o resultado pode ser conferido na Figura 3.7

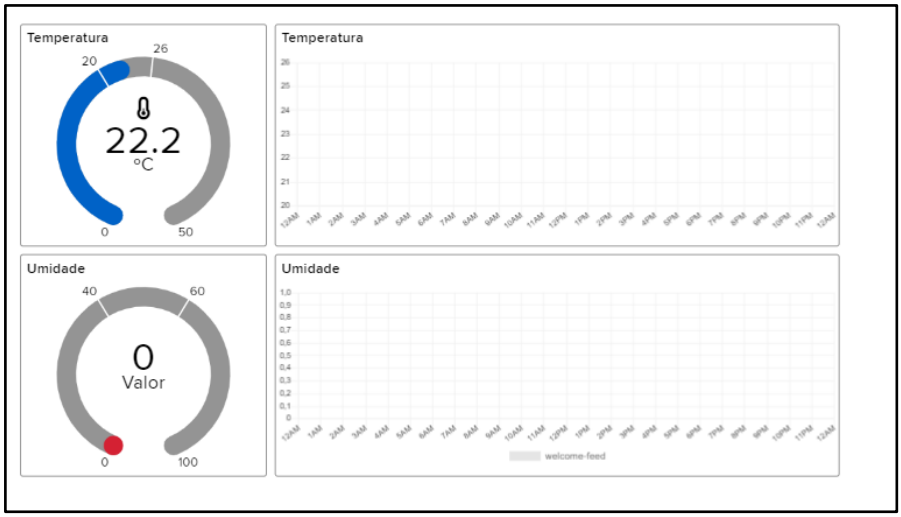
Figura 3.7 – Protoboard para teste do protótipo



Fonte: Autoria própria, 2023

Para concluir, é necessário desenvolver a interface IOT. Para isso, o primeiro passo é fazer o login no site Adafruit IO através de um computador. Em seguida, é preciso clicar na aba "Dashboards" e depois no botão "Add Dashboard". Nessa etapa, deve-se preencher os campos "Dashboard id", "Dashboard name" e "Dashboard description", respectivamente. Com as telas criadas, é possível inserir e configurar as ferramentas necessárias para controle e monitoramento do processo, como gráficos de temperatura e umidade, botões de controle e outros widgets disponíveis na plataforma. O objetivo é criar uma interface intuitiva e fácil de usar para visualização e controle dos dados coletados pelo sistema.

Figura 3.8 – Dashboard interface IOT



Fonte: Autoria própria, 2023

Após a montagem do protótipo e a realização da programação do ESP32, é necessário realizar o upload do sketch para o microcontrolador e verificar se os dados estão sendo enviados corretamente para a plataforma Adafruit IO.

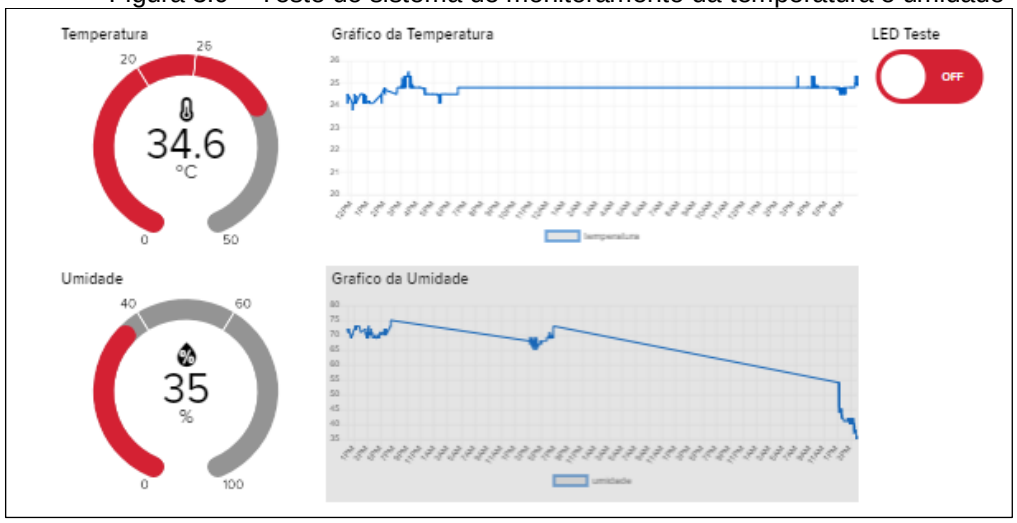
Para realizar o upload do sketch, é necessário conectar o cabo USB ao ESP32 e ao computador. Em seguida, é preciso selecionar a placa correta e a porta COM no menu Ferramentas da IDE do Arduino. Feito isso, basta clicar em “Enviar” para fazer o upload do sketch.

Após o upload, é possível testar o sistema de monitoramento. Para isso, é necessário verificar se os dados de temperatura e umidade estão sendo publicados corretamente na plataforma Adafruit IO. É possível visualizar os dados na dashboard criada anteriormente.

Também é importante verificar se a reconexão automática do ESP32 está funcionando corretamente. Para testar essa funcionalidade, é possível desligar e ligar o roteador ou desconectar e reconectar o cabo Ethernet. O ESP32 deve ser capaz de se reconectar à plataforma Adafruit IO automaticamente após a perda de conexão.

Com esses testes realizados com sucesso, o sistema de monitoramento utilizando o ESP32 e o sensor DHT11 funcionará corretamente e será possível monitorar a temperatura e a umidade de um determinado ambiente de forma remota e automatizada.

Figura 3.9 – Teste do sistema de monitoramento da temperatura e umidade



Autoria própria, 2023

3.5 Controle PID e controle

Para implementar o controle PID para uma lâmpada, utilizando um ESP32, um sensor de temperatura e umidade, o DHTxx e um dispositivo de controle de intensidade como um dimmer.

Conecte o ESP32 ao sensor de temperatura e ao dispositivo de controle de intensidade, o dimmer. Instale a biblioteca PID_v1.h no IDE Arduino, acessando o menu "Sketch" -> "Incluir Biblioteca" -> "Gerenciar Bibliotecas". Pesquise por "PID Library" (PID_v1) e clique em "Instalar" para adicionar a biblioteca ao seu ambiente de desenvolvimento.

No código, inclua a biblioteca PID_v1.h e defina as constantes e variáveis necessárias para o controle PID. Isso inclui a definição dos pinos utilizados pelo sensor de temperatura e pelo dispositivo de controle de intensidade, além do valor

desejado de temperatura (setpoint) e dos ganhos proporcional, integral e derivativo do controlador PID, conforme ilustra a Figura 3.10:

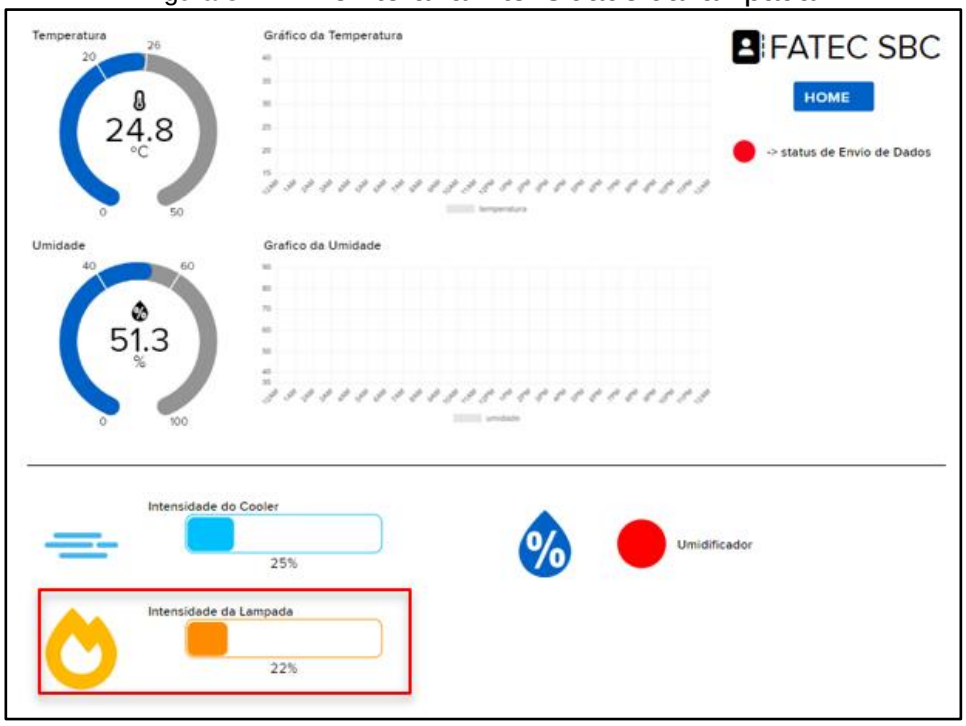
Figura 3.10 – Exemplo de controle PID

```
/********************************************************
 * PID Basic Example
 * Reading analog input 0 to control analog PWM output 3
 ********************************************************/

#include <PID_v1.h>

#define PIN_INPUT 0
#define PIN_OUTPUT 3

//Define Variables we'll be connecting to
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters
double Kp=2, Ki=5, Kd=1;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup()
{
  //initialize the variables we're linked to
  Input = analogRead(PIN_INPUT);
  Setpoint = 100;

  //turn the PID on
  myPID.SetMode(AUTOMATIC);
}

void loop()
{
  Input = analogRead(PIN_INPUT);
  myPID.Compute();
  analogWrite(PIN_OUTPUT, Output);
}
```

Autoria própria, 2023

Teste a integração com o Adafruit IO para monitorar a intensidade da lâmpada em tempo real. conforme ilustra a Figura 3.11:

Figura 3.11 – monitorar a intensidade da lâmpada

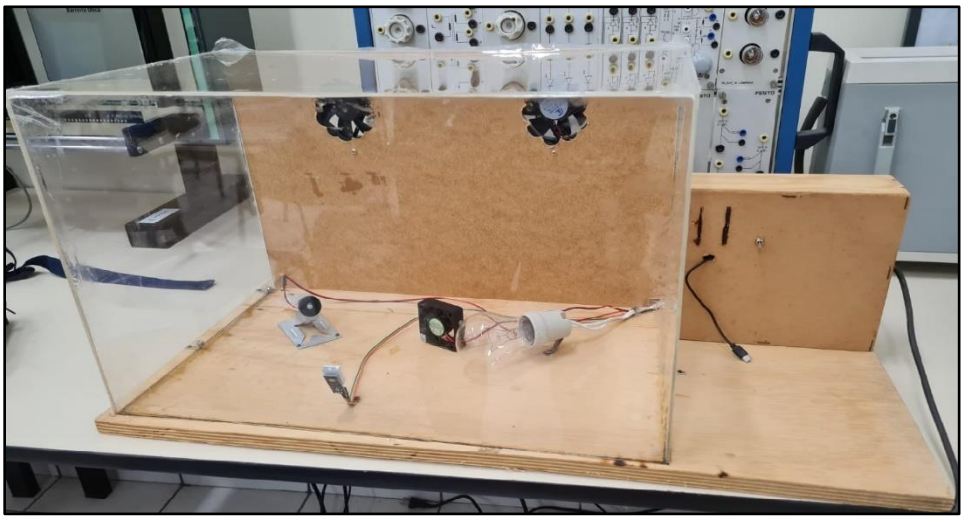


Autoria própria, 2023

3.6 Construção da estrutura do protótipo e esquema elétrico

Após os testes realizados de toda a parte eletrônica, realizamos a integração de todos os componentes no protótipo estrutural, decidimos fazer a base e a parte traseira em madeira para fixação dos componentes e todo o restante da estrutura em acrílico para uma melhor visualização conforme figura 3.12.

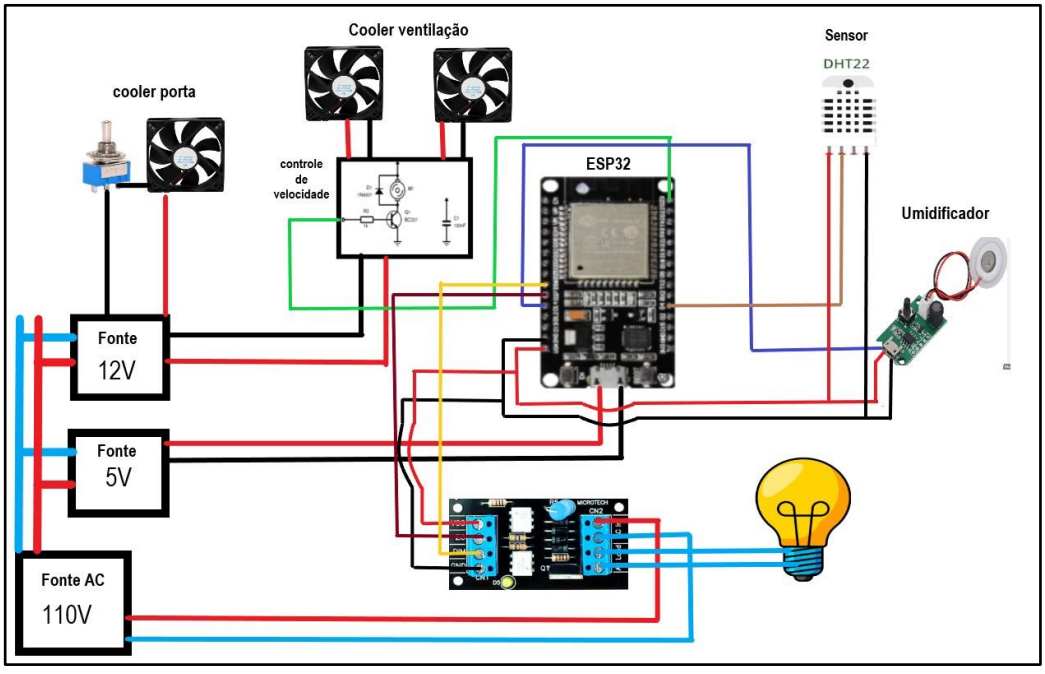
Figura 3.12 – Montagem da estrutura



Autoria própria, 2023

Para uma melhor visualização de toda a montagem elétrica, realizamos um esquema elétrico de todo o protótipo conforme figura 3.13.

Figura 3.13 – Esquema elétrico



Autoria própria, 2023

Após decisão do layout, componentes fixados e ligações elétricas feitas, realizamos todos os testes necessários do protótipo como um todo e validamos o seu funcionamento perfeitamente.

Deste modo conclui-se o desenvolvimento e construção do protótipo e desenvolvimento do algoritmo com sucesso.

CONSIDERAÇÕES FINAIS

O projeto intitulado Ambiente com Temperatura e Umidade Controlada para Salas SMD por meio da IOT, tem como objetivo desenvolver um ambiente com temperatura e umidade controlada por meio de sensores, atuadores e algoritmos, além da publicação e supervisão dos dados na nuvem. Tal objetivo foi concluído com êxito.

O desenvolvimento do projeto ocorreu no microcontrolador ESP32, com o auxílio de sensores e atuadores para o controle das grandezas físicas além da supervisão e relatórios na plataforma Adafruit IO.

Dentre as fontes pesquisadas necessárias para a execução do projeto, as que mais agregaram conhecimento foram sobre comunicação em nuvem, programação e integração dos sistemas no ESP32.

Os métodos e técnicas realizadas conforme a metodologia científica deram suporte para organizar as ideias e planejar a execução das etapas para o desenvolvimento do projeto. Deste modo foi possível desenvolver um sistema para o controle das grandezas físicas necessárias em salas SMD, onde esse mesmo conceito poderá ser aplicado para o controle de outros tipos de ambientes.

O projeto trouxe para o grupo conhecimentos em programação no ESP32, aplicação de conceitos de integração dos componentes com a nuvem e conhecimento de novos componentes eletrônicos.

Encontramos diversos desafios ao longo do projeto, porém conseguimos solucionar com a ajuda do corpo docente e embasamento teórico encontrado.

Conclui-se que o projeto foi desenvolvido com êxito dentro do esperado, trazendo diversos conhecimentos ao longo das etapas de montagem. Propõem como ações de melhorias futuras, mais componentes que atrapalhe o controle do nosso

monitoramento, a fim de mostrar os valores sendo normalizados dentro do set-point selecionado, além de uma parte estrutural mais elaborada.

REFERÊNCIAS

AOSONG. **datasheet DTH11.** Aosong Electronics Co. Ltd ,2016. 1 p.

AOSONG. **datasheet DTH22.** Aosong Electronics Co. Ltd, 2016. 2 p.

AUTOCORE. **Diferença entre os sensores DTH11 E DTH22.** 2017. Disponível em: <https://autocorerobotica.blog.br/diferencas-entre-os-sensores-dht11-e-dht22-2/>. Acesso em 18 de out. 2022.

BASUMALLICK, Chiradeep. **What Is MQTT (MQ Telemetry Transport)? Working, Types, Importance, and Applications.** 2022. Disponível em: <https://www.spiceworks.com/tech/iot/articles/what-is-mqtt/> Acesso em 31 de maio. 2023.

DAMASIO, Soraya. **Controle PID de Forma Simples e Descomplicada.** 2020. Disponível em: < https://www.citisystems.com.br/controle-pid/> Acesso em 31 de maio. 2023.

ESPRESSIF**. ESP32 Datasheet.** Shanghai: Espressif Systems, 2018. 10 p.

ESSS. **Artigos técnicos ano 2008**. Disponível em: <https://www.esss.co/blog/ predicao-de-falhas-em-componentes-eletronicos/>. Acesso em 26 ago. 2022.

GOUVEIA, Marcelo. **Etapas para a fabricação em SMT.** 2018. Disponível em: <https://produza.ind.br/tecnologia/fabricacao-em-smt>. Acesso em 10 out. 2022.

J. S. Vimala, M. Natesan, S. Rajendran. **Corrosion and Protection of Electronic Components in Different Environmental Conditions - An Overview.** The Open Corrosion Journal, 2009.

KERSCHBAUMER, R. et al. (2017). **Microcontroladores.** Universidade Federal

Catarinense. Câmpus Luzerna: IFC,2017.

MANUAL DE NORMALIZAÇÃO DE TCC – TRABALHO DE CONCLUSÃO DE CURSO. **Material didático para utilização nos projetos de trabalho de graduação, dos cursos de Tecnologia em Automação Industrial, Manufatura Avançada e Tecnologia em Informática para Negócios.** São Bernardo do Campo: Fatec, 2021.

MATTEDE, Henrique. **Componentes SMD, tipos e características.** 2019. Disponível em: <https://www.mundodaeletrica.com.br/componentes-smd-o-que-sao-tipos-caracteristicas>. Acesso em 10 de out. 2022.

MICHELL. **Controle da umidade relativa e temperatura em fábricas de produtos eletrônicos.** 2018. Disponível em: <http://www.michell.com/br/documents/appnotes/Controle_da_umidade_relativa_e_temperatura_em_f%C3%A1bricas_de_produtos_eletr%C3%B4nicos.pdf>. Acesso em 03 nov. 2022.

MOTT, Anderson. **Artigo controle de processo de 02/06/2021.** Disponível em: <https://www.automacaoindustrial.info/o-que-sao-sistemas-supervisorios/>. Acesso em 25/10/2022

OASIS**. MQTT Specifications.** Disponível em: <https://mqtt.org/mqtt-specification/> Acesso em 31 de maio. 2023.

OLIVEIRA, Sergio. **Componentes SMD, O que são? Tipos e características!** , **Internet da Coisas com ESP8266, Arduino e Raspberry Pi. ed.1**, São Paulo: Novatec, 2017.

PRODANOV, C.C.; FREITAS, E.C. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho científico. 2. ed. Rio Grande do Sul: Universidade Feevale, 2013.

Santos, S. (2017). **Porque o controle de temperartura é importante.** Disponível em: <https://www.linkedin.com/pulse/porque-o-controle-de-temperatura-e-umidade-na-%C3%A9-sidney-santos/?originalSubdomain=pt>. Acesso em 01 de nov. 2022.

SEVERINO, A. J. **Metodologia do trabalho científico**, 23. ed. São Paulo: Cortez, 2017.

SILVEIRA, C. B. **Saiba tudo Sobre Protocolo Modbus,** Disponível em**: <**https://www.citisystems.com.br/modbus/> Acesso em 25 de out. 2022.

TECHTUDO. **O que é internet das coisas? Veja como funciona IoT e exemplos de uso.** 2022. Disponível em: 1<https://www.techtudo.com.br/noticias/2022/10/o-que-e-internet-das-coisas-veja-como-funciona-a-iot-e-exemplos-de-uso.ghtml>. Acesso em 05 de out. 2022.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. de. **Sensores industriais – fundamentos e aplicações**. 4. ed. São Paulo: Editora Érica Ltda, 2012.

THERMOMATIC. **O que são supervisórios**. Disponível em: <https://www.thermomatic.com.br/fique-por-dentro/o-que-sao-sistemas-supervisorios.html> Acesso em 02 de abr. 2023.

APENDICE A – BIBLIOTECAS E VARIÁVEIS GLOBAIS

```
// Bibliotecas
#include <PID_v1.h>                    // Importa a biblioteca PID para cálculo do controle PID
#include <WiFi.h>                      // Inclui a biblioteca para comunicação WiFi
#include <DHT.h>                       // Inclui a biblioteca para o sensor DHT
#include "Adafruit_MQTT.h"             // Inclui a biblioteca para comunicação MQTT com o serviço
Adafruit IO
#include "Adafruit_MQTT_Client.h"      // Inclui a biblioteca para o cliente MQTT do Adafruit IO
#include <Arduino.h>                   // Inclui a biblioteca do ambiente de desenvolvimento do Arduino
#include <analogWrite.h>

//Variaveis Globais
#define DHTPIN 4                    // Define o pino do sensor DHT
//#define DHTTYPE DHT11             // Define o tipo do sensor DHT11
#define DHTTYPE DHT22               // Define o tipo do sensor DHT22

#define LED 2                       // Define o pino do LED embutido do ESP32
#define Umidificador 14             // Pino do Umidificador
#define Setpoint_Umidade_Ideal 50   // Valor de referência da umidade 50%
#define Umidade_Min 40              // Valor de referência da umidade mínima 40%
#define Umidade_ref 49.8            // Valor de referência da umidade 49.8%

#define COOLER_PIN 23               // Pino do cooler
#define SETPOINT 23.3               // Temperatura de setpoint Cooler
#define MAX_TEMP_GAP 4              // Diferença máxima para atingir 100% de capacidade
#define TEMP_STEP 0.3               // Incremento de temperatura para mudar a velocidade

DHT dht(DHTPIN, DHTTYPE);           // Cria uma instância do sensor DHT

float UltimaTemperatura = 0.0;      // Variável para armazenar a última temperatura lida
float UltimaUmidade = 0.0;          // Variável para armazenar a última umidade lida
int ultimo_speed = 0;
int ultima_intensidade = 0;

const int LedLoop = 2;              // Pino do LED para cada vez que fizer um loop acender
const int zc = 27;                  // Define o pino de entrada do detector de cruzamento por zero
const int dim = 26;                 // Define o pino de saída do circuito PWM
const int freq = 60;                // Define a frequência do sinal PWM

//const int botaoPorta = 22;        // Pino do botão

// Define as constantes do PID
double kp = 3;                      // Constante proporcional do PID
double ki = 6;                      // Constante integral do PID
double kd = 2;                      // Constante derivativa do PID
double Setpoint, Input, Output;     // Define as variáveis de entrada, saída e ponto de referência do PID
```

APENDICE B – CONFIGURAÇÃO DE CONEXÃO ENTRE O MICROCONTROLADORE E AO IDAIFRUIT IO

```
#define WIFI_SSID "WIFI"                          // Define o SSID da rede WiFi
#define WIFI_PASS "SENHA"                         // Define a senha da rede WiFi
#define AIO_SERVER "io.adafruit.com"              // Define o endereço do servidor do Adafruit IO
#define AIO_SERVERPORT 1883                       // Define a porta do servidor MQTT do Adafruit IO
#define AIO_USERNAME "nome usuario"               // Define o nome de usuário do Adafruit IO
#define AIO_KEY "chave API "                      // Define a chave de API do Adafruit IO


WiFiClient client;                        // Cria uma instância do cliente WiFi
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
// Cria uma instância do cliente MQTT do Adafruit IO

Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Temperatura");              // Cria uma instância do tópico MQTT para publicação de temperatura

Adafruit_MQTT_Publish umidade = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/Umidade");
// Cria uma instância do tópico MQTT para publicação de umidade

Adafruit_MQTT_Publish umidificador = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Umidificador");             // Cria uma instância do tópico MQTT para publicação sobre o estado do
Umidificador

Adafruit_MQTT_Publish lampada = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Lampada_Aquecedor");  // Cria uma instância do tópico MQTT para publicação sobre o estado da
Lampada

Adafruit_MQTT_Publish cooler = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Cooler_Ventilacao");       // Cria uma instância do tópico MQTT para publicação sobre o estado do
Cooler

Adafruit_MQTT_Publish Alarme = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/ Alarmes");
// Cria uma instância do tópico MQTT para publicação sobre Alarmes

Adafruit_MQTT_Publish RelatorioTime = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
Relatorio");                        // Cria uma instância do tópico MQTT para publicação sobre data hora
```

APENDICE C – FUNÇOES CONECTA E RECONECTAR

```
// Função para conectar ao WiFi e ao Adafruit IO
void connect() {
  Serial.print("Conectando ao WiFi...");       // Imprime na porta serial a mensagem de conexão ao WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASS);      // Inicia a conexão ao WiFi com o SSID e a senha definidos

  while (WiFi.status() != WL_CONNECTED) {     // Verifica se a conexão WiFi foi estabelecida
    delay(1000);                                          // Aguarda 1 segundo antes de verificar novamente
    Serial.print(".");                                    // Imprime um ponto para indicar que ainda está
conectando
  }

  Serial.println();          // Imprime uma quebra de linha para indicar que a conexão WiFi foi estabelecida
  Serial.println("Conectado ao WiFi com sucesso!");      // Imprime na porta serial que a conexão WiFi foi
estabelecida com sucesso

  Serial.print("Conectando ao Adafruit IO...");  // Imprime na porta serial a mensagem de conexão ao
Adafruit IO
  int8_t ret;                 // Cria uma variável para armazenar o retorno da conexão MQTT



// Loop principal que tenta se conectar ao servidor MQTT do Adafruit IO e se não conseguir, tenta
novamente após 5 segundos
  while ((ret = mqtt.connect()) != 0) {
    Serial.println(mqtt.connectErrorString(ret));                    // Imprime o erro de conexão na porta serial
    Serial.println("Tentando novamente em 5 segundos...");  // Imprime uma mensagem na porta serial
    mqtt.disconnect();                                                    // Desconecta do servidor MQTT
    delay(5000);                                                           // Espera 5 segundos novamente
  }

  Serial.println("Conectado ao Adafruit IO com sucesso!");   // Imprime uma mensagem na porta serial
quando a conexão é bem sucedida
}



// Função que tenta reconectar ao servidor MQTT do Adafruit IO
void reconnect() {
  // Loop que tenta reconectar ao servidor enquanto a conexão não é bem sucedida
  while (!mqtt.connected()) {
    Serial.println("Tentando reconectar ao Adafruit IO...");        // Imprime uma mensagem na porta serial
    connect();                      // Chama a função "connect" para tentar se conectar novamente

    // Se a conexão for bem sucedida, publica as últimas leituras de temperatura e umidade
    if (mqtt.connected()) {
      temperatura.publish(UltimaTemperatura);
      umidade.publish(UltimaUmidade);
    }

    delay(5000);                  // Espera 5 segundos antes de tentar se conectar novamente
  }
}
```

APENDICE D – FUNÇÃO UMIDIFICADOR

```
// Declare uma variável global para armazenar o estado anterior do umidificador
int previousState = -1;              // Inicialize com um valor inválido

void controlarUmidificador(float Umidade) {
  int currentState = -1;             // Variável para armazenar o estado atual do umidificador

  if (Umidade >= Setpoint_Umidade_Ideal) {
    digitalWrite(Umidificador, HIGH);                        // Desliga o umidificador
    Serial.println("Estado do Umidificador: Desligado");    // Exibe o estado do umidificador no monitor serial
    currentState = 0;                                        // Atribui 0 ao estado atual
  } else if (Umidade >= Umidade_Min && Umidade <= Umidade_ref) {
    digitalWrite(Umidificador, LOW);                        // Liga o umidificador
    Serial.println("Estado do Umidificador: Ligado");       // Exibe o estado do umidificador no monitor serial
    currentState = 1;                                        // Atribui 1 ao estado atual
  }

  // Verifica se o estado atual é diferente do estado anterior
  if (currentState != previousState) {
    // Atualiza o estado anterior com o estado atual
    previousState = currentState;

    // Realiza a publicação
    umidificador.publish(String(currentState).c_str());
  }
}
```

APENDICE E – FUNÇÃO LÂMPADA PID

```
PID myPID(&Input, &Output, &Setpoint, kp, ki, kd, DIRECT);          // Cria um objeto PID com as constantes definidas e o modo de operação direto


void LampadaControle() {
  float temp = dht.readTemperature();              // Lê a temperatura do sensor DHT
  float hum = dht.readHumidity();                  // Lê a umidade do sensor DHT

  if (isnan(temp) || isnan(hum)) {                 // Verifica se os dados lidos do sensor DHT são válidos
    Serial.println("Falha ao ler dados do sensor DHT11");
    return;
  }

  Input = temp;                                    // Define a entrada do PID como a temperatura lida
  myPID.Compute();                                 // Executa o cálculo do PID

  // Imprime os valores de temperatura, umidade e saída do PID
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.print(" °C, Umidade:");
  Serial.print(hum);
  Serial.print(", Intensidade: ");
  Serial.println(Output);

  if (ultima_intensidade != Output) {
    lampada.publish(Output);
    ultima_intensidade = Output;  // Atualiza o valor da última intensidade
  }

  int dutyCycle = map(Output, 0, 100, 51, 255);     // Mapeia a saída do PID para um valor de duty cycle
  ledcWrite(1, dutyCycle);                         // Escreve o valor de duty cycle no canal 1 do circuito PWM

  digitalWrite(LED, HIGH);                              // Acende o LED embutido do ESP32
  delay(750);
  digitalWrite(LED, LOW);                               // Apaga o LED embutido do ESP32
  delay(750);
}
```

APENDICE F – FUNÇÃO COOLER

```
void controlCooler(float temperaturaAtual) {
  static int previousSpeed = -1;                        // Variável para armazenar a velocidade anterior

  if (temperaturaAtual > SETPOINT) {
    int speed = map(temperaturaAtual, SETPOINT, SETPOINT + MAX_TEMP_GAP, 0, 100);
    speed = constrain(speed, 0, 100);                   // Limita a velocidade entre 0 e 100

    if (speed != previousSpeed) {                       // Verifica se a velocidade é diferente do valor anterior
      analogWrite(COOLER_PIN, map(speed, 0, 100, 0, 255));
      Serial.print("Velocidade do cooler: ");
      Serial.print(speed);
      Serial.println("%");
      cooler.publish(speed);
      previousSpeed = speed;                            // Atualiza o valor da velocidade anterior
    }
  } else {
    analogWrite(COOLER_PIN, 0);
    Serial.println("Cooler desligado");
    if (previousSpeed != 0) {                           // Verifica se a velocidade é diferente do valor anterior
      cooler.publish(0);
      previousSpeed = 0;                                // Atualiza o valor da velocidade anterior
    }
  }

  if (temperaturaAtual > SETPOINT + TEMP_STEP) {
    int tempDivisions = (temperaturaAtual - SETPOINT) / TEMP_STEP;
    int speed = map(tempDivisions, 0, (MAX_TEMP_GAP / TEMP_STEP), 0, 100);
    speed = constrain(speed, 0, 100);                   // Limita a velocidade entre 0 e 100

    if (speed != previousSpeed) {                       // Verifica se a velocidade é diferente do valor
anterior
      analogWrite(COOLER_PIN, map(speed, 0, 100, 0, 255));
      previousSpeed = speed;                            // Atualiza o valor da velocidade anterior
    }
  } else {
    analogWrite(COOLER_PIN, 0);
    if (previousSpeed != 0) {                           // Verifica se a velocidade é diferente do valor anterior
      previousSpeed = 0;                                // Atualiza o valor da velocidade anterior
    }
  }
}
```

APENDICE G – VOID SETUP

```
void setup() {
  Serial.begin(115200);            // Inicia a comunicação serial com uma taxa de transmissão de 115200
bps
  delay(10);                                  // Aguarda 10ms

  dht.begin();                                // Inicializa o sensor DHT11
  connect();                                  // Chama a função para conectar ao WiFi e ao Adafruit IO

  pinMode(Umidificador, OUTPUT);              // Define o pino do umidificador como saída
  digitalWrite(Umidificador, HIGH);           // Inicialmente, desliga o Umidificador
  pinMode(zc, INPUT);                         // Define o pino de entrada do detector de cruzamento por
zero como entrada
  pinMode(dim, OUTPUT);                       // Define o pino de saída do circuito PWM como saída
  pinMode(LED, OUTPUT);                       // Define o pino do LED embutido do ESP32 como saída


  ledcAttachPin(dim, 1);                      // Associa o pino de saída do circuito PWM ao canal 1
  ledcSetup(1, freq, 8);                      // Configura o canal 1 do circuito PWM com a frequência e
resolução definidas


  // Define o valor ideal do sensor para o controle PID
  Setpoint = 25.8;

  // Configura o PID/
  myPID.SetMode(AUTOMATIC);                   // Configura o modo de operação do PID como
automático
  myPID.SetOutputLimits(20, 100);             // Define os limites de saída do PID como 0 e 100
  pinMode(COOLER_PIN, OUTPUT);
}
```

APENDICE H – VOID LOOP

```
void loop() {

  if (!mqtt.connected()) {
    reconnect();                  // Chama a função para reconectar ao Adafruit IO se a conexão for perdida
  }
  mqtt.processPackets(3500);          // Processa os pacotes MQTT a cada 3.5 segundos
  mqtt.ping();                        // Envia um sinal de ping para manter a conexão MQTT ativa

  float temperaturaAtual = dht.readTemperature();            // Lê a temperatura atual do sensor DHT11
  float umidadeAtual = dht.readHumidity();                   // Lê a umidade atual do sensor DHT11

  if (isnan(temperaturaAtual) || isnan(umidadeAtual)) {        // Verifica se as leituras são válidas
    Serial.println("Falha ao ler o sensor DHT11!");             // Imprime uma mensagem de falha na leitura
na na porta serial
    Alarme.publish("Falha ao ler o sensor DHT11!");           // Imprime uma mensagem de falha na leitura
na na portal mqtt
    return;                                                   // Retorna ao início do loop
  }
  if (temperaturaAtual != UltimaTemperatura || umidadeAtual != UltimaUmidade) {     // Verifica se a
temperatura ou umidade mudou desde a última leitura
    UltimaTemperatura = temperaturaAtual;                      // Atualiza a última temperatura lida
    UltimaUmidade = umidadeAtual;                             // Atualiza a última umidade lida
    RelatorioTime.publish("->");
    temperatura.publish(temperaturaAtual);                    // Publica a nova temperatura no tópico MQTT
    umidade.publish(umidadeAtual);                            // Publica a nova umidade no tópico MQTT

    if(temperaturaAtual>26.0){          // Verifica se a Temperatura ideal está acima do setpoint desejado
      Alarme.publish("Temperatura está alta!");         // Publica que a temperatura está acima do setpoint
    }
    else if(temperaturaAtual<20.0){    // Verifica se a Temperatura ideal está abaixo do setpoint desejado
      Alarme.publish("Temperatura está baixa!");       // Publica que a temperatura está abaixo do setpoint
    }
    if(umidadeAtual>60.0){               // Verifica se a Umidade ideal está acima do setpoint desejado
      Alarme.publish("Umidade  está alta!");             // Publica que a Umidade está acima do setpoint
    }
    else if(umidadeAtual<40.0){                          // Verifica se a Umidade  ideal está abaixo do
setpoint desejado
      Alarme.publish("Umidade  está baixa!");            // Publica que a Umidade  está abaixo do setpoint
    }

    controlarUmidificador(umidadeAtual); // Chama a função para controlar o Umidificador com base na
umidade
    LampadaControle();                                   // Chama a função para controlar a lampada com
base na temperatura
    controlCooler(temperaturaAtual);  // Chama a função para controlar o cooler com base na temperatura
  }
  digitalWrite(LedLoop, HIGH);                  // Acende o LED indicador
  delay(500);                                   // Aguarda 500ms
  digitalWrite(LedLoop, LOW);                   // Apaga o LED indicador
  delay(500);                                   // Aguarda 500ms segundos
}
```