Yuri Siqueira Dantas

Desenvolvimento de um Sistema de Monitoramento Ambiental com ESP32 e Sensores de Gás e Umidade integrado ao Blynk

Campina Grande, PB

2024

Yuri Siqueira Dantas

Desenvolvimento de um Sistema de Monitoramento Ambiental com ESP32 e Sensores de Gás e Umidade integrado ao Blynk

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Orientador: Prof. Dr. Jaidilson Jó da Silva

Campina Grande, PB

2024

Yuri Siqueira Dantas

Desenvolvimento de um Sistema de Monitoramento Ambiental com ESP32 e Sensores de Gás e Umidade integrado ao Blynk

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, PB, _____/_____/_____.

Prof. Dr. Jaidilson Jó da Silva
Orientador

Gutemberg Gonçalves dos Santos Júnior
Convidado

Campina Grande, PB
2024

Dedico este trabalho à minha família, namorada e amigos, pelo apoio incondicional e por estarem sempre ao meu lado em cada etapa desta jornada.

Agradecimentos

Agradeço primeiramente a Deus, por estar ao meu lado ao longo desta jornada, sustentando-me nos momentos difíceis e guiando meus passos. Agradeço também a Santa Maria, por interceder em minhas preces, trazendo-me conforto e esperança.

Sou eternamente grato aos meus pais, que me proporcionaram não apenas o suporte necessário para completar esta etapa, mas também amor, apoio incondicional e inspiração. A eles devo tudo o que sou e conquistei até aqui. Ao meu irmão, por ser um grande companheiro de jornada, e à minha namorada, pelo constante apoio nos momentos mais desafiadores, sempre me encorajando a seguir em frente e oferecendo seu companheirismo nas horas de incerteza. Aos meus avós, que, com tanto amor e incentivo, me motivaram a continuar.

Agradeço também ao meu orientador, professor Jaidilson Jó da Silva, por suas valiosas orientações e pela confiança depositada em mim e no meu trabalho.

Aos amigos que caminharam comigo, especialmente aos companheiros do grupo "SSG", que estiveram presentes virtualmente durante as longas madrugadas de estudo; aos amigos do grupo "Lake", que sempre me inspiraram a me tornar um engenheiro; aos parceiros do grupo "Rô", que tornaram a aventura da graduação mais leve; e aos amigos do grupo "Equinos", pela amizade, companheirismo e fraternidade ao longo dessa caminhada.

A todos vocês, minha eterna gratidão por fazerem parte desta jornada, que se tornou mais leve e significativa graças ao apoio de cada um. Obrigado por tudo!

"Enquanto não lutares contra a frivolidade, a tua cabeça será semelhante a uma loja de objetos velhos: não conterá senão utopias, sonhos e trastes velhos." (São Josemaria Escrivã)

Lista de ilustrações

Lista de abreviaturas e siglas

Lista de Abreviações

ADC	Conversor Analógico para Digital (Analog-to-Digital Converter)
AES	Padrão de Criptografia Avançada (Advanced Encryption Standard)
BLE	Bluetooth de Baixo Consumo (Bluetooth Low Energy)
CO	Monóxido de Carbono
CO_2	Dióxido de Carbono
CPU	Unidade Central de Processamento (Central Processing Unit)
DAC	Conversor Digital para Analógico (Digital-to-Analog Converter)
GPIO	Entrada/Saída de Uso Geral (General-Purpose Input/Output)
IR	Infravermelho (Infrared)
IoT	Internet das Coisas (Internet of Things)
kHz	Quilohertz
MHz	Megahertz
nm	Nanômetro
OLED	Diodo Orgânico Emissor de Luz (Organic Light-Emitting Diode)
PM	Material Particulado (Particulate Matter)
PWM	Modulação por Largura de Pulso (Pulse Width Modulation)
SRAM	Memória de Acesso Aleatório Estática (Static Random-Access Memory)
NO_2	Dióxido de Nitrogênio
O_3	Ozônio Troposférico
SO_2	Dióxido de Enxofre
µA	Microampere
VOC	Compostos Orgânicos Voláteis (Volatile Organic Compounds)
Wi-Fi	Rede Sem Fio (Wireless Fidelity)

Resumo

As mudanças climáticas globais e eventos recentes, como queimadas intensas, têm aumentado a frequência e a gravidade de situações que afetam diretamente a qualidade do ar e a saúde humana. As populações vulneráveis, como idosos, crianças e pessoas com doenças respiratórias, são as mais afetadas. Evidenciando a importância do monitoramento contínuo da qualidade do ar e das condições climáticas internas, destacando a necessidade de sistemas que possam fornecer dados precisos em tempo real para uma resposta rápida e eficaz. Este trabalho apresenta o desenvolvimento de um sistema de monitoramento ambiental baseado na Internet das Coisas (IoT), que utiliza o microcontrolador ESP32, sensores DHT11 e MQ-135 para monitorar variáveis como temperatura, umidade e qualidade do ar em ambientes internos. O projeto busca contribuir para a promoção de ambientes mais saudáveis, permitindo a detecção precoce de condições ambientais adversas e facilitando a intervenção imediata.

Palavras chave: Saúde humana, Monitoramento contínuo, Condições climáticas internas, Dados precisos em tempo real, Internet das Coisas (IoT).

Abstract

Global climate changes and recent events, such as intense wildfires, have increased the frequency and severity of situations that directly affect air quality and human health. Vulnerable populations, such as the elderly, children, and people with respiratory diseases, are the most affected. This highlights the importance of continuous monitoring of air quality and indoor climatic conditions, emphasizing the need for systems that can provide accurate, real-time data for a quick and effective response. This work presents the development of an environmental monitoring system based on the Internet of Things (IoT), which uses the ESP32 microcontroller, DHT11, and MQ-135 sensors to monitor variables such as temperature, humidity, and air quality in indoor environments. The project aims to contribute to promoting healthier environments, enabling early detection of adverse environmental conditions and facilitating immediate intervention.

Keywords: Human health, Continuous monitoring, Indoor climatic conditions, Accurate real-time data, Internet of Things (IoT)

Sumário

1 Introdução

A qualidade do ar é fundamental para a saúde e o bem-estar das pessoas, e seus impactos são particularmente graves em populações vulneráveis, como recém-nascidos, crianças, idosos e indivíduos com condições de saúde pré-existentes. Devido às suas características fisiológicas únicas, essas populações são mais suscetíveis aos efeitos nocivos da exposição a poluentes atmosféricos, que podem resultar em uma variedade de problemas respiratórios e cardiovasculares. Crianças, por exemplo, possuem um sistema respiratório em desenvolvimento e taxas de respiração mais elevadas em relação ao peso corporal, o que as torna mais vulneráveis aos poluentes presentes no ar (MAYNARD et al., 1999). A exposição prolongada durante a infância pode ter consequências de longo prazo, como a redução da função pulmonar.

Diante disso, a necessidade de monitoramento da qualidade do ar em ambientes internos é crescente, especialmente em locais com alta densidade populacional e poluição. Ambientes com má qualidade do ar podem desencadear problemas de saúde como asma, alergias e desconforto geral, comprometendo a qualidade de vida das pessoas (SPENGLER; SAMET; MCCARTHY, 2000). Assim, a gestão eficaz da qualidade do ar interior é essencial para assegurar ambientes saudáveis e promover o bem-estar de todos.

Com o avanço da tecnologia da Internet das Coisas (IoT), surgem novas oportunidades para o desenvolvimento de soluções de monitoramento da qualidade do ar que sejam acessíveis, eficientes e intuitivas. A disseminação dos sistemas de microcontroladores, como o Arduino e o ESP32, tem desempenhado um papel crucial nesse cenário, proporcionando plataformas de fácil utilização que permitem tanto a amadores quanto a profissionais a criação de projetos de IoT de maneira ágil e prática. Essa acessibilidade tem facilitado a implementação de tecnologias sofisticadas para o monitoramento ambiental em uma variedade de contextos.

Grande parte dessa popularidade deve-se ao baixo custo associado a esses microcontroladores, que oferecem uma plataforma versátil para a criação de soluções tecnológicas que, antes, demandavam conhecimentos avançados em eletrônica e programação. A ampla comunidade de desenvolvedores e a vasta disponibilidade de bibliotecas e tutoriais também contribuem significativamente para a democratização do uso dessas tecnologias, tornando-as acessíveis para projetos em áreas como automação residencial, saúde e monitoramento ambiental (MONK; MCCABE, 2016).

1.1 Objetivos

Desenvolver um sistema de monitoramento de qualidade do ar utilizando uma ESP32 integrada a um aplicativo móvel, com o propósito de capturar dados de temperatura, umidade e concentração de gases. O sistema deve enviar essas informações para o aplicativo, onde o usuário poderá definir níveis de alerta personalizados, configurar parâmetros para a automatização do ar-condicionado e do umidificador, além de acompanhar o histórico das medições. O objetivo é oferecer uma ferramenta prática e acessível que permita monitorar e controlar a qualidade do ar de forma contínua, contribuindo para a prevenção de problemas de saúde associados à poluição e garantindo um ambiente mais saudável e confortável.

1.2 Organização do Trabalho

O trabalho está estruturado em 5 capítulos, incluindo este introdutório, conforme a seguir.

O **Capítulo 2** Fundamentação Teórica, expondo os conceitos teóricos para a contextualização e entendimento do trabalho desenvolvido.

O **Capítulo 3** aborda os materiais e metodologias utilizadas para o desenvolvimento do trabalho bem como também a integração das partes.

No **Capítulo 4** são apresentados os resultados dos testes realizados.

O **Capítulo 5** apresenta as conclusões do trabalho, expondo os resultados alcançados, assim como uma análise dos pontos falhos e propostas futuras visando complementar as atividades desenvolvidas.

2 Fundamentação Teórica

Neste capítulo, será realizada uma revisão teórica abrangente sobre os principais conceitos e tecnologias que sustentam o desenvolvimento deste sistema de monitoramento ambiental. Serão abordados os fundamentos relacionados à qualidade do ar e seus impactos na saúde, o funcionamento dos sensores ambientais utilizados, e as capacidades do microcontrolador ESP32 no contexto da IoT. Além disso, serão exploradas as tecnologias de comunicação infravermelha (IR) e o uso de relés para automação de dispositivos, como ar-condicionado e umidificador. Por fim, será discutido o papel das plataformas de IoT na criação de interfaces gráficas e no gerenciamento remoto, destacando a importância da automação para o controle eficiente de variáveis ambientais e a promoção de ambientes mais saudáveis e sustentáveis.

2.1 Qualidade do Ar e Seus Impactos na Saúde

A qualidade do ar é determinada pela concentração de poluentes nocivos no ambiente, sendo mensurada por parâmetros estabelecidos por agências de saúde, como a Organização Mundial da Saúde (OMS). Entre os principais poluentes atmosféricos estão o material particulado (PM), dióxido de nitrogênio (NO_2), dióxido de enxofre (SO_2), monóxido de carbono (CO) e ozônio troposférico (O_3), que podem afetar gravemente a saúde humana e o meio ambiente. De acordo com a OMS, a exposição prolongada a esses agentes está associada ao aumento de doenças respiratórias crônicas e cardiovasculares, especialmente em populações vulneráveis, como crianças e idosos (World Health Organization, 2021). O monitoramento da qualidade do ar também leva em consideração fatores como temperatura e umidade, que influenciam a dispersão dos poluentes em ambientes internos, onde gases como dióxido de carbono (CO_2) e compostos orgânicos voláteis (VOCs) podem indicar riscos adicionais à saúde.

Nas grandes metrópoles brasileiras, como São Paulo e Manaus, a qualidade do ar tem sido uma crescente preocupação. Em São Paulo, fatores como o aumento da poluição industrial, tráfego veicular e inversões térmicas frequentes durante o inverno contribuem para a concentração elevada de poluentes, especialmente material particulado, frequentemente ultrapassando os limites recomendados pela OMS (CETESB - Companhia Ambiental do Estado de São Paulo, 2023). Já em Manaus, o problema é agravado pelas queimadas na Amazônia, que, em combinação com a seca severa causada pelo El Niño, resultaram na terceira pior qualidade do ar do mundo em outubro de 2023, afetando diretamente a saúde da população (FEIFEL, 2023).

Estudos indicam que, entre os poluentes atmosféricos, o material particulado fino

(PM2.5) é particularmente nocivo à saúde, por penetrar profundamente nos pulmões e na corrente sanguínea, aumentando o risco de doenças respiratórias e cardiovasculares, como infartos e derrames. Outros poluentes, como óxidos de nitrogênio (NO_x) e SO_2, intensificam esses efeitos, contribuindo para o aumento de internações hospitalares e mortes prematuras (BRUNEKREEF; HOLGATE, 2002; POPE; DOCKERY, 2006). Este cenário reforça a necessidade de políticas públicas eficazes que controlem a emissão de poluentes e mitiguem seus impactos na saúde pública.

Portanto, a qualidade do ar é um indicador essencial da saúde e do bem-estar populacional. Em regiões urbanas densamente povoadas, onde a poluição pode alcançar níveis alarmantes, monitorar e controlar as condições ambientais é fundamental para proteger a saúde pública. A implementação de ferramentas tecnológicas para o monitoramento em tempo real e a análise contínua da qualidade do ar pode auxiliar na formulação de políticas públicas e estratégias de mitigação mais eficazes.

2.2 Sensores Ambientais Utilizados no Monitoramento da Qualidade do Ar

Nesta seção, será abordado o funcionamento dos principais sensores utilizados no monitoramento da qualidade do ar, focando nas tecnologias envolvidas, nos princípios de operação e nas suas aplicações práticas no sistema proposto. Os sensores que serão discutidos incluem o DHT11, utilizado para medir temperatura e umidade, e o MQ135, responsável pela detecção de gases nocivos no ambiente.

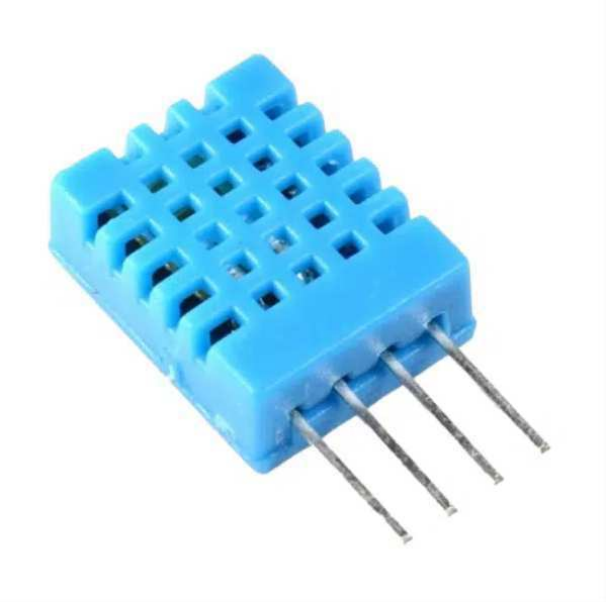
2.2.1 Sensor de Temperatura e Umidade: DHT11

O DHT11 é um sensor amplamente utilizado para medir temperatura e umidade relativa em projetos de baixo custo e de fácil implementação. Ele opera com um termistor resistivo para medir a temperatura e um sensor capacitivo para medir a umidade. O sensor capacitivo de umidade detecta a variação da capacitância causada pela umidade no ar, gerando um sinal que é convertido em valores digitais. O DHT11 tem uma precisão de ±2°C para a temperatura e de ±5% para a umidade (MONK; MCCABE, 2016; ADAFRUIT, 2020), o que o torna adequado para monitoramento ambiental básico. Seu intervalo de operação é de 0°C a 50°C para temperatura e de 20% a 90% para umidade relativa, cobrindo uma ampla gama de aplicações em ambientes internos e externos.

O sensor DHT11 se comunica com microcontroladores por meio de um protocolo digital, utilizando apenas um fio de dados. Ele é uma opção ideal para o monitoramento em tempo real, permitindo a captura de dados de forma contínua com baixa demanda de energia e fácil integração a plataformas IoT, como a ESP32. Apesar de o DHT11 ser

menos preciso que sensores como o DHT22 ou o BME280, ele foi escolhido devido ao seu custo-benefício e facilidade de integração ao sistema proposto.

Figura 1 – Imagem do sensor de temperatura e umidade DHT11.



Fonte: (MAKERHERO, 2023)

2.2.2 Sensor de Gases: MQ135

O MQ135 é um sensor de gás de baixo custo, capaz de detectar uma ampla gama de gases nocivos, como amônia (NH_3), óxidos de nitrogênio (NO_x), dióxido de carbono (CO_2) e compostos orgânicos voláteis (VOCs) (ELECTRONICS, 2016; LIBELIUM, 2017). O sensor opera com base em um princípio eletroquímico, onde a resistência interna do sensor varia conforme a concentração de gases presentes no ar. Essa mudança de resistência é lida como um sinal analógico, que pode ser convertido em valores digitais para interpretação dos níveis de poluição.

Nesse contexto, esse sensor possui uma faixa de detecção que pode variar de 10 a 1000 ppm (partes por milhão) dependendo do gás específico, além de ser frequentemente utilizado em sistemas de monitoramento da qualidade do ar devido à sua capacidade de detectar múltiplos gases, o que o torna uma escolha ideal para ambientes urbanos e industriais. No sistema proposto, o MQ135 desempenha um papel crucial ao alertar o usuário sobre a presença de gases perigosos em níveis insalubres.

Figura 2 – Imagem do sensor de gases MQ135.

Fonte: (MAKERHERO, 2022)

2.2.3 Integração dos Sensores no Sistema

A integração dos sensores DHT11 e MQ135 ao sistema de monitoramento de qualidade do ar é feita utilizando a ESP32 como controlador central. O DHT11 envia leituras digitais de temperatura e umidade, enquanto o MQ135 fornece leituras analógicas da concentração de gases. Com isso, são processadas pela ESP32 e enviadas para um aplicativo móvel, onde o usuário pode visualizar os dados em tempo real, definir níveis de alerta personalizados e acompanhar o histórico das medições.

A combinação desses sensores permite um monitoramento eficaz da qualidade do ar, oferecendo dados cruciais para o controle ambiental. nesse cenário, o DHT11 e o MQ135 foram escolhidos por sua confiabilidade, baixo custo e facilidade de implementação em projetos de IoT, que visam o monitoramento contínuo e acessível de variáveis ambientais.

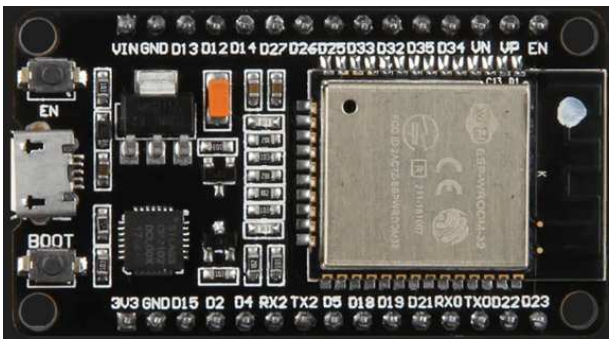
2.3 Microcontroladores e IoT no Monitoramento Ambiental

Os microcontroladores desempenham um papel central em sistemas IoT, com o ESP32 sendo um dos mais versáteis e populares. Suas capacidades de comunicação Wireless Fidelity (Wi-Fi) e Bluetooth permitem sua utilização em projetos que envolvem a coleta de dados de sensores e o envio dessas informações para plataformas remotas. A seguir, será discutido como o ESP32 é empregado no monitoramento ambiental e como o conceito de IoT transforma o acompanhamento de parâmetros como qualidade do ar e condições climáticas.

2.3.1 O Microcontrolador ESP32

O ESP32 é um microcontrolador de baixo custo e alta performance desenvolvido pela Espressif Systems, que integra um microprocessador de 32 bits com conectividade Wi-Fi e Bluetooth. Suas capacidades de comunicação sem fio permitem que ele se conecte a redes de Internet para transmitir e receber dados de forma remota, o que é essencial em sistemas de monitoramento. Ele possui um conjunto robusto de interfaces, incluindo General-Purpose Input/Output (GPIOs) para conexão com sensores, Pulse Width Modulation (PWM), Analog-to-Digital Converter (ADC), Digital-to-Analog Converter (DAC) e Serial Peripheral Interface (SPI), o que o torna uma plataforma poderosa para a automação e controle de dispositivos.Essas interfaces permitem ao ESP32 trabalhar com uma variedade de sensores e atuadores, possibilitando sua aplicação em diferentes cenários de automação, como o controle de dispositivos elétricos e o monitoramento remoto de variáveis ambientais.

Figura 3 – Imagem do microcontrolador ESP32.



Fonte: (MARKER, 2024)

No contexto do monitoramento ambiental, o ESP32 é frequentemente utilizado para coletar dados de sensores, como o DHT11 e o MQ135, e enviar essas informações para plataformas IoT em nuvem. A sua comunicação Wi-Fi permite que ele envie dados diretamente para esses serviços, facilitando a visualização em tempo real e o armazenamento de históricos para futuras análises. Esses recursos o tornam uma solução versátil e eficaz em projetos de monitoramento contínuo, tanto em aplicações domésticas quanto em ambientes industriais. Comparado a modelos anteriores como o ESP8266, o ESP32 oferece maior poder de processamento e conectividade mais avançada, tornando-o ideal para projetos que demandam multitarefa e alta eficiência no consumo de energia (MONK; MCCABE, 2016).

Deve-se destacar que, o ESP32, por conta de sua arquitetura dual-core, permite a execução de múltiplas tarefas simultâneas, com um desempenho eficiente tanto para aplicações de baixa potência quanto para projetos que exigem processamento intensivo. Esse microcontrolador é alimentado por um processador Tensilica Xtensa LX6, capaz de atingir velocidades de até 240 Megahertz (MHz), além de oferecer modos de baixa

energia, como o deep sleep, que reduz significativamente o consumo energético, sendo ideal para aplicações em dispositivos IoT que operam com baterias. Estudos e benchmarks demonstram que o ESP32 se posiciona como uma escolha confiável e eficiente para sistemas de monitoramento e automação, superando alternativas em termos de consumo energético e processamento em tempo real (MAHETALIYA et al., 2021).

No modo deep sleep, o ESP32 pode reduzir seu consumo para menos de 10 Microampere (µA), tornando-o especialmente útil em dispositivos que precisam operar por longos períodos com uma única carga de bateria (SYSTEMS, 2019). Outra característica notável é seu suporte para criptografia via hardware, que permite a implementação de algoritmos como Advanced Encryption Standard (AES), garantindo maior segurança na transmissão de dados em redes Wi-Fi, o que é essencial em projetos que lidam com informações sensíveis, como monitoramento de saúde ou segurança residencial.

2.3.2 Comunicação e conectividade

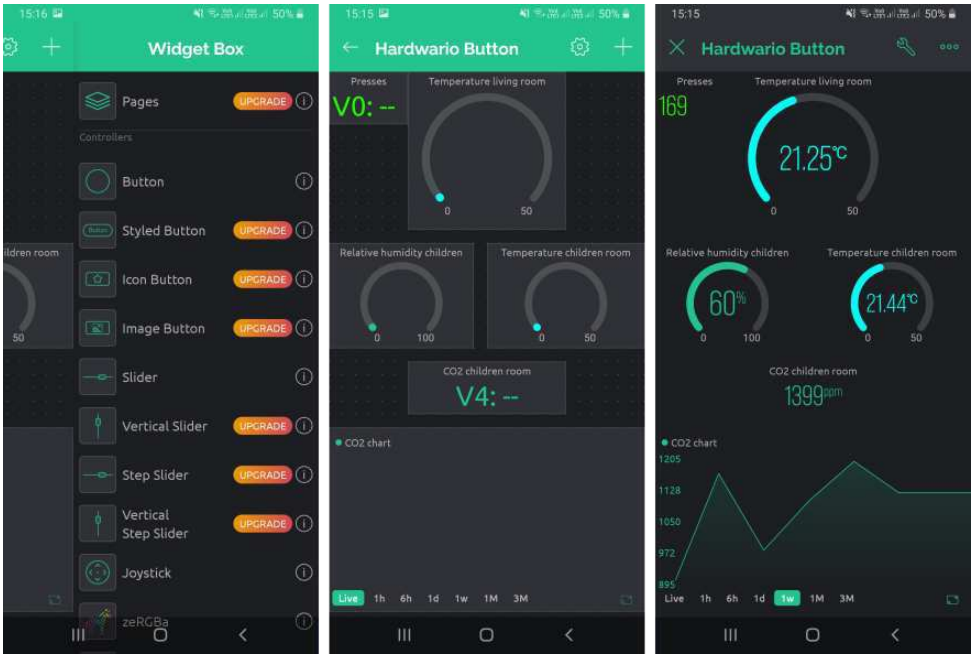
O Wi-Fi integrado no ESP32 permite sua conexão a redes sem fio, atuando tanto como cliente quanto como Access Point, dependendo da necessidade do sistema. No modo acess point, ele pode criar uma rede local, permitindo que outros dispositivos se conectem diretamente, sem a necessidade de uma infraestrutura Wi-Fi existente. Isso é útil em cenários de monitoramento remoto, onde não há conexão de internet disponível (SYSTEMS, 2019). Além disso, o Bluetooth Low Energy (BLE) do ESP32 é altamente eficiente em termos de consumo de energia, sendo ideal para comunicação de curto alcance com dispositivos móveis, como smartphones, para configuração rápida ou ajustes locais.

A comunicação sem fio do ESP32 é suportada por sua unidade central de processamento (CPU) dual-core e memória estática de acesso aleatório (SRAM), que garantem processamento eficiente, mesmo com múltiplas tarefas rodando simultaneamente. sse conjunto de características faz com que esse microcontrolador seja uma plataforma confiável e flexível para aplicações IoT, onde a troca de dados em tempo real e a eficiência energética são essenciais (KALAIVANI et al., 2023).

2.3.3 O conceito de IoT

A Internet das Coisas (IoT) refere-se à interconexão de dispositivos físicos, como sensores e atuadores, que são capazes de coletar, processar e compartilhar dados pela internet (GUBBI et al., 2013). No contexto de monitoramento ambiental, a IoT permite que sistemas autônomos realizem medições contínuas de qualidade do ar, temperatura e umidade. Sensores conectados a plataformas como Blynk ou ThingSpeak facilitam a análise em tempo real desses dados, permitindo a automação de processos críticos, como o acionamento de sistemas de ventilação ou controle de poluição em tempo hábil.

Figura 4 – Interface do Blynk utilizada para monitoramento IoT.



Fonte: (DOCS, 2024)

A capacidade de fornecer dados em tempo real e automatizar respostas rápidas a condições ambientais adversas torna a IoT indispensável para a preservação da saúde pública e a proteção ambiental,possibilitando o uso mais eficiente de energia, com dispositivos como o ESP32 sendo capazes de operar em modos de baixo consumo, como o deep sleep, prolongando a vida útil de dispositivos alimentados por baterias (SYSTEMS, 2019).

2.4 Automação de Dispositivos com IR e Relés

A automação de dispositivos utilizando IR e relés desempenha um papel importante em sistemas de controle remoto e automação residencial. Essas tecnologias permitem o controle remoto de eletrodomésticos, sistemas de iluminação, ar-condicionado e outros dispositivos elétricos, proporcionando conforto e eficiência ao usuário. Neste contexto, o ESP32 atua como uma interface entre o controle via IR e o acionamento de relés, facilitando a automação de dispositivos de maneira econômica e eficiente.

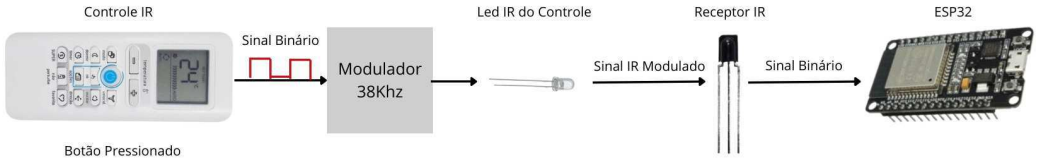
2.4.1 Funcionamento do Controle Infravermelho

O controle infravermelho é amplamente utilizado para o acionamento de dispositivos eletrônicos, como televisores, ar-condicionado e sistemas de som. O controle via IR utiliza pulsos de luz invisíveis ao olho humano (na faixa dos 850-950 nm) para transmitir comandos a um receptor IR no dispositivo. Esses comandos são modulados em uma frequência

específica, normalmente 38 kHz, pois essa modulação é controlada por temporizadores, que garantem a precisão do sinal e evitam interferências.

O ESP32, combinado com um emissor e receptor IR (como o TSOP38238), pode capturar e reproduzir os sinais IR de controles remotos, permitindo que o microcontrolador "aprenda"os comandos enviados. A biblioteca IRremote é frequentemente utilizada para capturar e enviar esses sinais com precisão, decodificando diferentes padrões de protocolos, como NEC, Sony, e RC5 (MONK; MCCABE, 2016). Cada protocolo possui características próprias de codificação de dados, com o NEC utilizando uma codificação por pulso espaçado e o Sony codificação por comprimento de pulso.

Figura 5 – Diagrama da modulação do sinal IR



Fonte: Autoria Própria

Uma vez que o ESP32 é capaz de ler e reproduzir esses sinais IR, ele pode ser programado para controlar dispositivos como ar-condicionado e televisores com base em condições ambientais monitoradas, por exemplo, ligando um aparelho quando a temperatura ambiente ultrapassa um certo nível.

2.4.2 Uso de Relés na Automação

O relé é um componente eletromecânico que funciona como um interruptor controlado eletronicamente, permitindo a automação de dispositivos de maior potência. Podendo ser utilizado para ligar ou desligar dispositivos que operam em alta tensão, a partir de um sinal de baixa tensão gerado pelo microcontrolador.

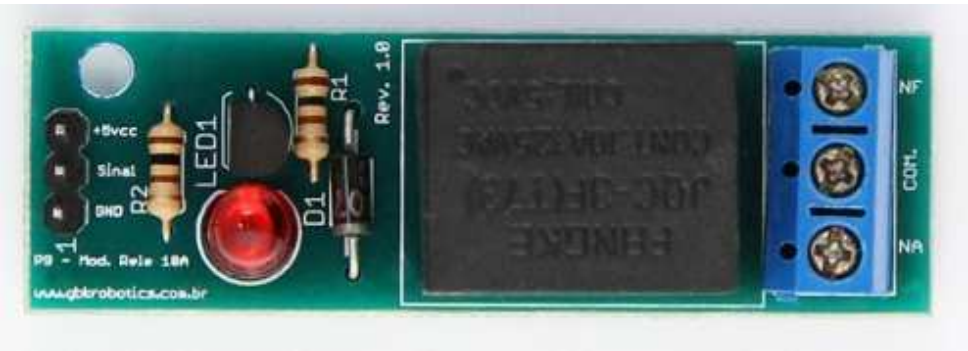
Geralmente possui três pinos de entrada e três terminais de saída. Os pinos de entrada são:

- VCC: É o pino de alimentação, ele fornece a energia necessária para acionar o circuito interno. Relés geralmente operam com uma tensão de controle de 3.3V ou 5V o que permite uma interface direta com o ESP32.
- GND: Este pino é conectado ao terra (GND) do sistema, criando o circuito fechado necessário para o funcionamento do relé.
- IN (entrada) ou sinal: O pino de controle, onde o microcontrolador envia o sinal para acionar o relé.

Os três terminais de saída são:

- COM (Comum): É o terminal comum de saída, que se conecta a um dos lados do dispositivo que você deseja controlar.
- NA (Normalmente Aberto): Este terminal se mantém aberto, quando o relé é acionado, o NA é fechado, permitindo que a corrente flua entre ele e o COM, permitindo assim que o dispositivo ligado ao relé através dessa saída seja acionado.
- NF (Normally Fechado): Este terminal é o oposto do NA; o circuito entre COM e NF permanece fechado enquanto o relé está desativado. Ao acionar o relé, este circuito é aberto, desligando o dispositivo que esteja ligado atráves dessa saída.

Figura 6 – Imagem do módulo relé de saída única



Fonte: Adaptado de Eletru's

O ESP32 pode controlar um ou mais relés, acionando-os conforme comandos recebidos remotamente ou parâmetros pré-programados, como a temperatura e a qualidade do ar. Por exemplo, o ESP32 pode monitorar a qualidade do ar com o sensor MQ135 e acionar um relé para ligar um ventilador ou purificador de ar quando a concentração de gases nocivos atinge um nível crítico (LIBELIUM, 2020).

2.4.3 Aplicações e Vantagens

O uso de IR e relés em conjunto com microcontroladores, como o ESP32, permite a criação de soluções eficientes para automação residencial e industrial. Entre as vantagens estão:

- Automação remota: o controle de dispositivos pode ser feito à distância, seja por meio de um controle IR, seja por comandos enviados pela Internet.
- Eficiência energética: o ESP32 pode controlar o acionamento de dispositivos com base em leituras de sensores, otimizando o uso de energia elétrica em sistemas de ventilação, aquecimento ou resfriamento.
- Versatilidade: a combinação de infravermelho e relés oferece uma ampla gama de aplicações, desde o controle de pequenos aparelhos até a automação de grandes sistemas de alta potência.

Com essa flexibilidade, o ESP32 e essas tecnologias são amplamente utilizados em projetos de automação residencial, sistemas de monitoramento de energia e em aplicações de IoT voltadas à sustentabilidade.

3 Metodologia de Pesquisa

O sistema desenvolvido tem como objetivo monitorar as condições ambientais de um local, controlando equipamentos com base nas informações coletadas, para garantir que o bem-estar do usuário seja alcançado de maneira autônoma, através do controle automatizado das variáveis ambientais. Com isso, o sistema utiliza sensores para medir temperatura, umidade e gases, além de realizar o controle de dispositivos como ar-condicionado e umidificadores, por meio de um emissor infravermelho e um relé. Tendo, por sua vez, a comunicação entre o usuário e o sistema feita através do software conectado ao microcontrolador, permitindo o monitoramento em tempo real via uma aplicação mobile.

Ademais, há a possibilidade de visualização direta dos dados no hardware, com o auxílio de uma tela OLED, e o usuário é alertado sobre condições adversas através de um buzzer, reforçando tanto a interatividade do sistema como sua capacidade de notificar mudanças imediatas. Com isso, promovendo a automação do ambiente de forma eficiente, onde o controle dos equipamentos se baseia na leitura dos sensores instalados. Dessa forma, o sistema garante que o ambiente permaneça confortável e seguro, otimizando a eficiência energética e melhorando a qualidade de vida dos usuários.

Neste capítulo, será apresentada uma descrição detalhada do projeto desenvolvido, abordando os recursos físicos e virtuais utilizados, assim como a integração entre esses elementos. O capítulo está dividido em duas seções principais: Materiais e Metodologia. Na seção Materiais, será apresentada a lista de componentes e ferramentas empregadas na construção do sistema e na seção Metodologia, será explicado como esses materiais foram utilizados para atingir os objetivos propostos.

3.1 Materiais

3.1.1 ESP32

O ESP32 é o microcontrolador central do projeto, responsável pela coleta de dados dos sensores, controle de dispositivos e comunicação via Wi-Fi e Bluetooth. Ele é amplamente utilizado em várias aplicações de IoT, graças ao seu baixo custo, versatilidade e confiabilidade. Suas múltiplas interfaces, tornam-no adequado para uma ampla gama de aplicações, desde automação residencial até monitoramento ambiental. Com capacidades robustas de processamento e conectividade, ele oferece a flexibilidade necessária para lidar com a coleta de dados em tempo real e o controle de dispositivos conectados.

3.1.2 DHT11

O DHT11 é um sensor com alta popularidade, principalmente devido ao seu baixo custo e facilidade de uso. Ele mede a temperatura e a umidade relativa do ambiente. Embora sua precisão seja de ±2°C para temperatura e ±5% para umidade, inferior a outros sensores disponíveis no mercado, ele ainda se mostra eficaz em projetos de monitoramento básico. É amplamente empregado em soluções onde o custo-benefício é crucial e a necessidade de precisão extrema não é essencial.

3.1.3 MQ135

O MQ135 é um sensor de gases utilizado para medir a concentração de substâncias nocivas em ambientes internos e externos. Este sensor se destaca por seu custo-benefício e popularidade, especialmente em sistemas de monitoramento da qualidade do ar. A resistência interna do sensor varia de acordo com a concentração de gases, o que permite que ele forneça uma leitura precisa dos níveis de poluição do ar. Para garantir a precisão e a consistência das medições, o sensor foi calibrado utilizando um código específico, ajustando sua sensibilidade conforme as condições do ambiente de teste.

3.1.4 Módulo Relé

O Módulo Relé SRD-5VDC-SL-C é um componente eletromecânico usado para controlar dispositivos de maior potência, funcionando como um interruptor controlado eletronicamente. Ele opera com uma tensão de controle de 5V fornecida pelo ESP32, permitindo o controle de dispositivos que operam em corrente alternada (AC) até 250V. Este relé é ideal para controlar disositivos através de uma aplicação programada, tornando-se uma ferramenta essencial para automação de dispositivos de alta potência.

Figura 7 – Imagem do módulo Relé SRD-5VDC-SL-C

Fonte: (Eletrogate, 2023b)

3.1.5 LED Emissor IR

O LED emissor infravermelho é utilizado para enviar sinais a dispositivos que operam com controle remoto, como televisores e ar-condicionado. Quando utilizado em conjunto com o ESP32, permite a automação destes dispositivos, facilitando o controle de equipamentos à distância.

Figura 8 – Imagem do LED Emissor IR

Fonte: (MakerHero, 2023a)

3.1.6 Módulo Receptor Infravermelho

O VS1838B é um receptor infravermelho de alta sensibilidade, capaz de detectar sinais IR em uma frequência de 38kHz. Ele é amplamente utilizado em projetos de automação que envolvem controlar remotamente equipamentos via IR.

Figura 9 – Imagem do módulo Receptor IR VS1838B

Fonte: (MakerHero, 2023b)

3.1.7 Buzzer

O buzzer ativo de 5V Bip Contínuo é um componente sonoro utilizado para emitir alertas. Ele possui a característica de funcionar automaticamente ao receber uma tensão de 5V, sem a necessidade de um controle de frequência externo. No sistema desenvolvido, o buzzer é utilizado para alertar o usuário sobre condições adversas no ambiente, como níveis elevados de gases perigosos. Sua simplicidade e eficiência o tornam adequado para sistemas de alerta em tempo real.

Figura 10 – Imagem do buzzer ativo 5V Bip Contínuo

Fonte: (Eletrogate, 2023a)

3.1.8 Módulo Display OLED 128x64

A tela OLED 128x64 é utilizada para exibir os dados coletados pelos sensores em tempo real. Este tipo de display é conhecido por sua alta definição e contraste, o que permite uma excelente visualização dos dados, mesmo em ambientes com pouca iluminação. Ela oferece uma interface visual eficiente para o usuário monitorar as condições ambientais diretamente no hardware, sem depender exclusivamente da aplicação mobile.

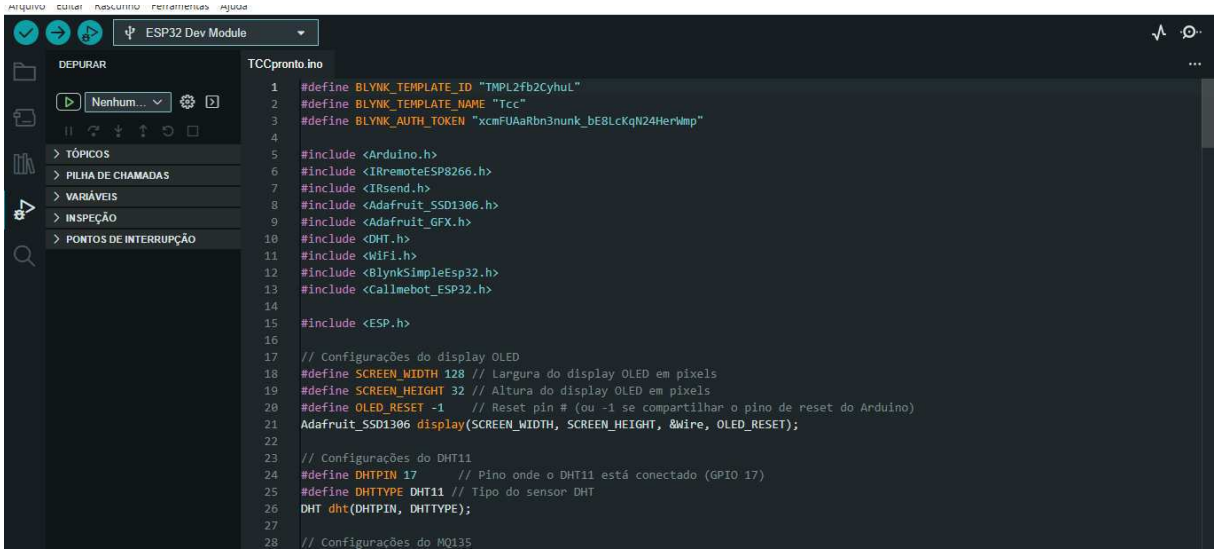
Figura 11 – Imagem do módulo Display OLED 128x64

Fonte: (RoboCore, 2023)

3.1.9 Arduino IDE

O Arduino IDE é a plataforma de desenvolvimento escolhida para a programação do ESP32 devido à sua simplicidade e eficiência. Este ambiente de desenvolvimento integrado permite a escrita, compilação e upload de códigos diretamente para o microcontrolador de forma intuitiva, facilitando o processo para desenvolvedores de todos os níveis. Além disso, o Arduino IDE oferece suporte a uma ampla variedade de bibliotecas, que simplificam a integração do ESP32 com sensores, dispositivos infravermelhos e plataformas de IoT. Sua interface amigável, combinada com a extensa documentação e comunidade ativa, proporciona uma vasta gama de recursos e tutoriais de fácil acesso, tornando-o uma ferramenta amplamente utilizada tanto em projetos amadores quanto profissionais.

Figura 12 – Codigo visualizado a partir da Arduino IDE



Fonte: Autoria Própria

3.1.10 Blynk IoT

O Blynk é uma plataforma IoT que facilita a comunicação entre dispositivos físicos, como o ESP32, e uma interface móvel. Com o Blynk, é possível monitorar e controlar os dispositivos do projeto em tempo real via um aplicativo móvel, sem a necessidade de desenvolver uma interface gráfica customizada. A plataforma fornece uma interface drag-and-drop intuitiva, permitindo ao usuário criar dashboards personalizados com diferentes widgets, como botões, sliders e displays de dados.

O Blynk permite a integração entre o ESP32 e a nuvem, possibilitando a coleta e análise de dados remotamente. Além disso, o Blynk possibilita a automação de tarefas, como o acionamento de dispositivos com base nos valores lidos pelos sensores, e o envio de alertas ao usuário quando certos limites são atingidos. A plataforma é amplamente utilizada em projetos de IoT por sua flexibilidade, eficiência e simplicidade de implementação.

Figura 13 – Interface do Blynk

Fonte: (Blynk, 2023)

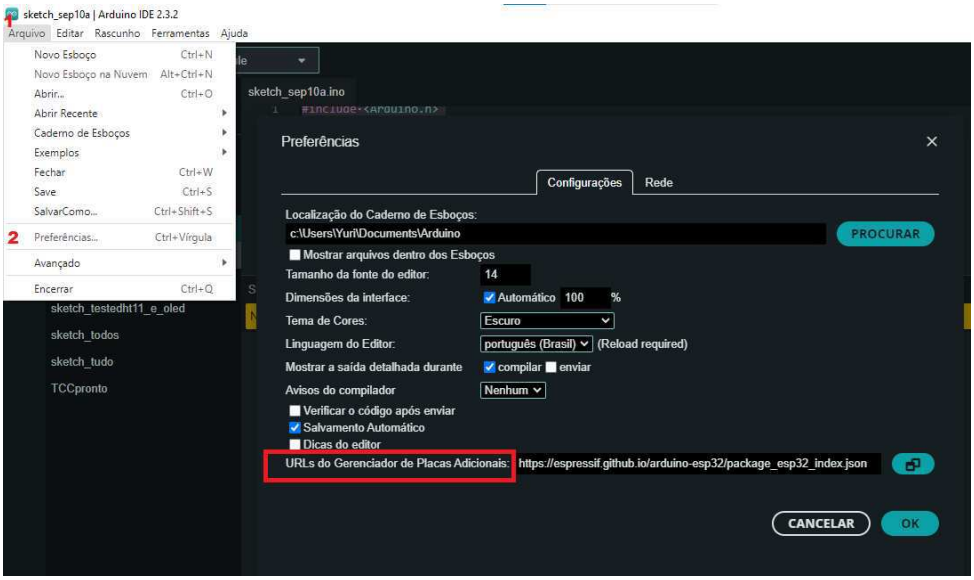
3.2 Metodologia

3.2.1 Ambiente de Desenvolvimento

O Arduino IDE foi a plataforma de desenvolvimento escolhida para a implementação do código no ESP32. Após a instalação, a primeira etapa para utilizar o ESP32 no Arduino IDE envolve configurar o ambiente de desenvolvimento para reconhecer o microcontrolador, para realizar essa configuração, é necessário adicionar o suporte à placa. O processo se inicia com a inclusão do link do gerenciador de placas do ESP32 nos parâmetros da IDE. Para isso, os seguintes passos foram seguidos:

1. No Arduino IDE clicamos em Arquivo > Preferências.
2. No campo **URLs do Gerenciador de Placas adicionais**, foi inserido o seguinte link: *https://espressif.github.io/arduino-esp32/package_esp32_index.json*

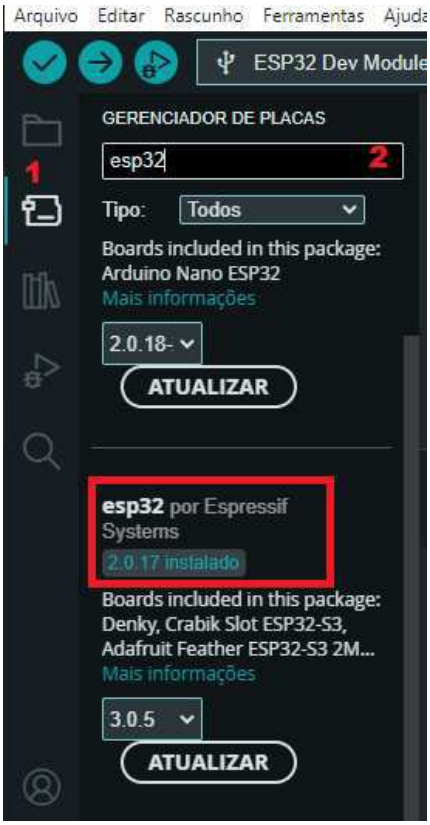
Figura 14 – Representação do passo a passo para configuração da ESP32 na Arduino IDE



Fonte: Autoria Própria

3. Em seguida, em "Gerenciador de Placas", presente no menu lateral da IDE, pesquisando por ESP32, foi instalado o pacote "ESP32 by Espressif Systems".

Figura 15 – Representação do passo a passo para instalação do pacote da ESP32 na Arduino IDE



Fonte: Autoria Própria

Uma vez instalado, no menu de seleção ao lado do botão de build, foi selecionado a placa ESP32 Dev Module, garantindo assim, que a IDE se comunique com o microcontrolador que será utilizado.

Figura 16 – Representação da placa ESP32 Dev Module selecionada na Arduino IDE

Fonte: Autoria Própria

3.2.2 Integração das Bibliotecas Utilizadas

Neste projeto, várias bibliotecas foram utilizadas para facilitar a integração dos sensores, dispositivos e comunicação via IoT. Cada uma delas desempenha um papel crucial no funcionamento e automação do sistema. A seguir, descrevemos quais foram as principais bibliotecas e suas funções no projeto.

3.2.2.1 IRremoteESP8266.h e IRsend.h

A biblioteca IRremoteESP8266.h é responsável pela recepção e transmissão de sinais infravermelhos utilizando o ESP32. O sistema depende dela para capturar sinais de controle remoto, como os enviados por um ar-condicionado, e para transmitir esses sinais, posteriormente, via o emissor infravermelho.

A IRsend.h é uma sub-biblioteca que, em conjunto com a IRremoteESP8266.h, cuida especificamente da emissão de sinais IR. Neste trabalho, ela foi utilizada para enviar o RAW Data capturado do controle remoto do ar-condicionado, permitindo que o sistema ligue ou desligue o aparelho automaticamente com base em leituras de temperatura e umidade.

Essas bibliotecas permitem que o ESP32 "aprenda"os comandos de dispositivos controlados por IR e os reproduza.

3.2.2.2 Adafruit_SSD1306.h e Adafruit_GFX.h

A biblioteca Adafruit_SSD1306.h é utilizada para controlar o display OLED 128x64, que exibe os dados dos sensores em tempo real. Através dela, o projeto pode mostrar informações como a temperatura, a umidade e a qualidade do ar diretamente no dispositivo, sem depender apenas da interface mobile.

A Adafruit_GFX.h complementa a SSD1306, oferecendo funções gráficas como a possibilidade de desenhar formas geométricas e textos customizados. Ela torna a interface do display mais versátil, permitindo a formatação e estilização dos dados exibidos. Isso melhora a usabilidade e a apresentação visual do sistema.

3.2.2.3 DHT.h

A biblioteca DHT.h é usada para capturar as leituras do sensor DHT11, que monitora a temperatura e a umidade do ambiente. Ela simplifica a leitura do sensor, convertendo os sinais brutos em dados facilmente utilizáveis pelo ESP32, que são então processados para determinar se o ambiente precisa de ajustes automáticos.

3.2.2.4 BlynkSimpleEsp32.h

A BlynkSimpleEsp32.h é a principal biblioteca para a integração com a plataforma Blynk IoT, pois ela permite que o ESP32 se conecte à internet e envie dados para a aplicação móvel Blynk, onde o usuário pode monitorar as condições do ambiente em tempo real, como a temperatura, a umidade e a qualidade do ar.

Além disso, a biblioteca possibilita que, através do uso do Blynk, seja possivel enviar sinais para a placa, tornando o sistema interativo. O BlynkSimpleEsp32.h facilita a

integração direta com o aplicativo mobile, permitindo a comunicação em tempo real e o armazenamento de históricos de dados.

3.2.2.5 Callmebot_ESP32.h

A biblioteca Callmebot_ESP32.h foi integrada ao projeto para o envio de alertas via WhatsApp através da API do CallMeBot. Esta funcionalidade permite que o sistema notifique o usuário sobre a detecção de níveis elevados de gases nocivos no ambiente. Ela foi especialmente útil na implementação de um sistema de notificação automático, aprimorando a interatividade do projeto e proporcionando uma forma adicional de comunicação sobre condições críticas no ambiente onde o hardware está instalado.

Para integrar essas bibliotecas ao Arduino IDE, foram realizados os seguintes passos:

1. Clicamos em Gerenciador de Bibliotecas no menu lateral da IDE;
2. No campo de busca, foi digitado o nome da biblioteca;
3. Verificada a biblioteca correspondente, foi realizado a instalação;
4. o processo foi repetido para todas as bibliotecas utilizadas.

Figura 17 – Representação do passo a passo para a inserção de biblioteca no Arduino IDE

Fonte: Autoria Própria

3.2.3 Testes com os Sensores e Dispositivos

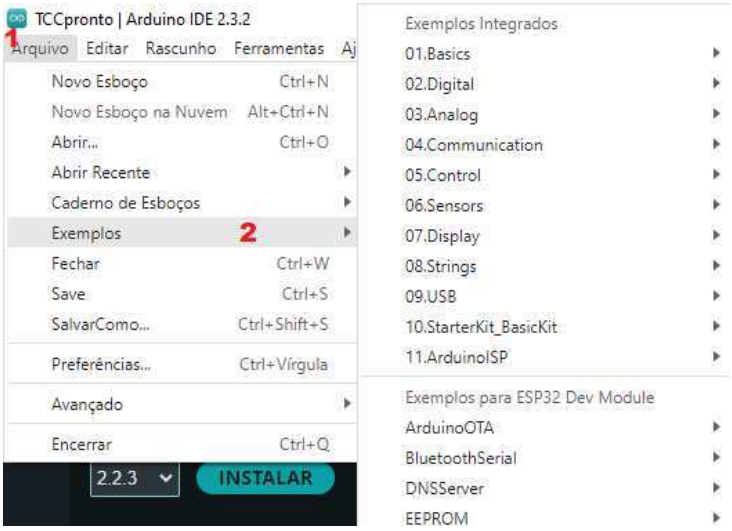
Para testar os sensores e garantir o correto funcionamento de todos os componentes utilizados no projeto, foram usados exemplos básicos disponíveis nas bibliotecas relacionadas a cada sensor. Esses exemplos são encontrados diretamente na interface do Arduino IDE, fornecendo uma base confiável para verificar a integridade dos sensores e dispositivos antes de sua integração ao código final.

Para Acessar os Exemplos no Arduino IDE, os passos seguidos foram:

1. Na barra de menus, selecionamos Arquivo > Exemplos.
2. No menu suspenso, foi selecionado a biblioteca desejada para o teste e logo em seguida o sketch adequado para realizar o teste

Diferentes skecths foram usados para testar todos os componente.

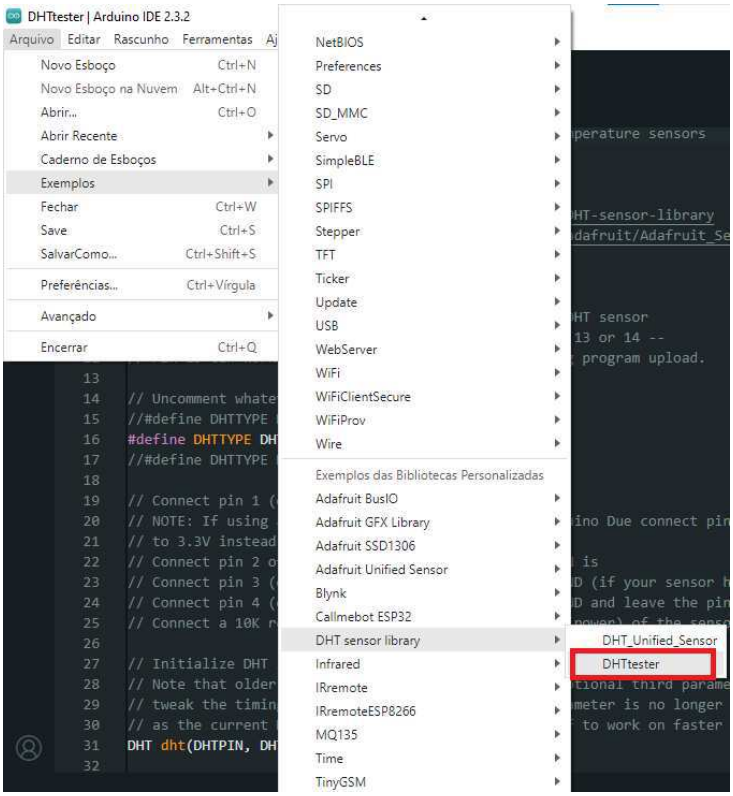
Figura 18 – Representação do passo a passo para utilização dos exemplos no Arduino IDE



Fonte: Autoria Própria

Para testar o DHT11, foi utilizado o exemplo DHTTester, que pode ser encontrado na biblioteca DHT. Este código simples lê e exibe a temperatura e umidade do ambiente no monitor serial, validando o correto funcionamento do sensor.

Figura 19 – Representação do caminho para o exemplo DHTTester no Arduino IDE



Fonte: Autoria Própria

O sensor MQ135 não precisa de uma biblioteca específica, mas foi testado utilizando um código simples de leitura analógica com a função analogRead().

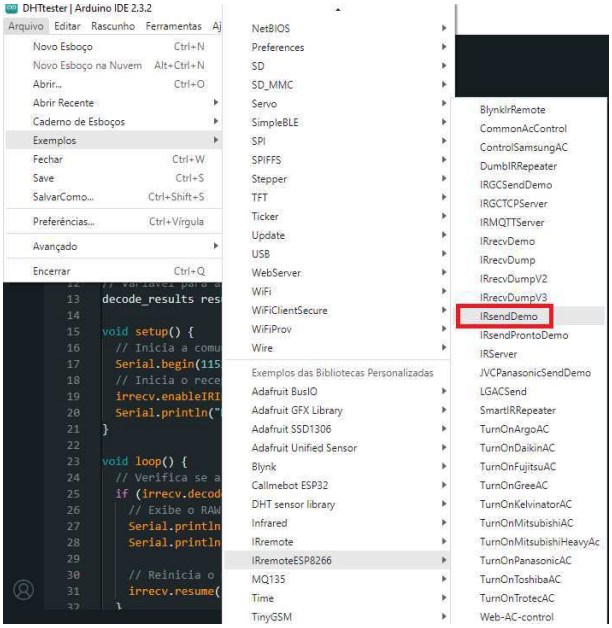
Figura 20 – Código utilizado para testar o MQ135

```
int mq135Pin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int gasLevel = analogRead(mq135Pin);
  Serial.print("MQ135 Gas Level: ");
  Serial.println(gasLevel);
  delay(1000);
}
```

Fonte: Autoria Própria

Para testar o emissor infravermelho, foi utilizado o exemplo IRSendDemo da biblioteca IRremoteESP8266.h.

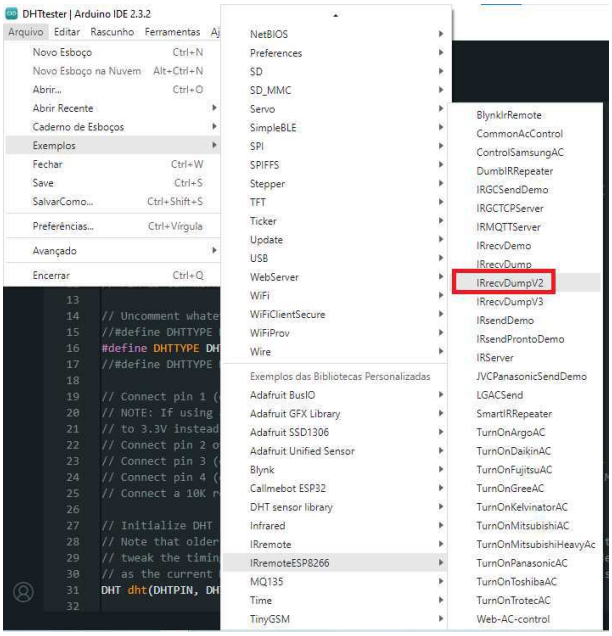
Figura 21 – Representação do caminho para o exemplo IRSendDemo no Arduino IDE



Fonte: Autoria Própria

Para capturar os sinais enviados por controles remotos, foi utilizado o exemplo IRrecvDumpV2, também encontrado na biblioteca IRremoteESP8266.h. Este exemplo permite capturar o RAW Data dos sinais IR enviados por qualquer controle remoto.

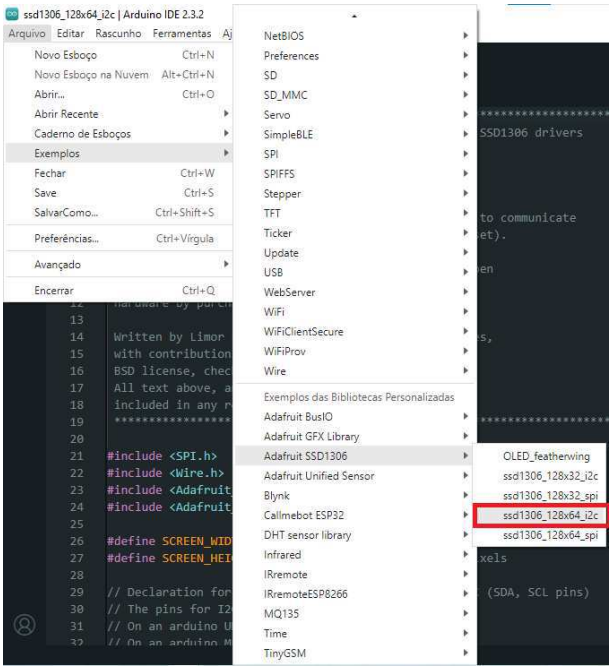
Figura 22 – Representação do caminho para o exemplo IRrecvDumpV2 no Arduino IDE



Fonte: Autoria Própria

O teste da tela OLED 128x64 foi realizado utilizando um exemplo básico fornecido pela biblioteca Adafruit_SSD1306. Para verificar a funcionalidade da tela, foi utilizado o exemplo ssd1306_128x64_i2c. Esse exemplo demonstra como inicializar a tela, exibir texto, e desenhar formas básicas, como linhas e retângulos, permitindo uma rápida verificação da integridade do display.

Figura 23 – Representação do caminho para o exemplo ssd1306_128x64_i2c. no Arduino IDE



Fonte: Autoria Própria

O uso desses exemplos básicos foi crucial para garantir que cada sensor e dispositivo estivesse funcionando corretamente antes de serem integrados ao sistema completo. Depois de confirmar o bom funcionamento de todos os componentes, capturamos os códigos infravermelhos responsáveis por ligar e desligar o ar-condicionado no ambiente onde o hardware seria testado.

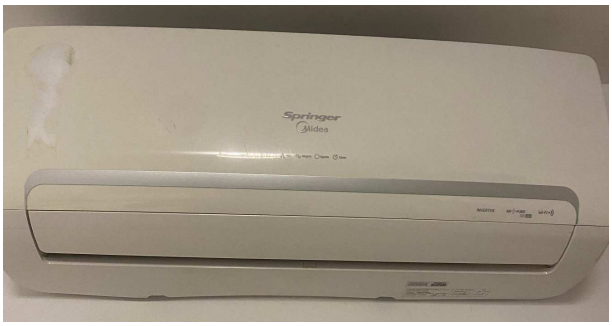
3.2.4 Captura do RawData do controle do ar-condicionado

O modelo de ar-condicionado utilizado para os testes de automação foi o Springer Midea, um equipamento amplamente popular em ambientes residenciais e comerciais. Esse modelo foi escolhido por sua disponibilidade no local dos testes e por operar com controle remoto via infravermelho, o que permitiu a captura dos sinais RawData correspondentes aos comandos de ligar e desligar.

Figura 24 – Fotografia do controle do ar-condicionado Springer Midea

Fonte: Autoria Própria

Figura 25 – Fotografia do ar-condicionado Springer Midea



Fonte: Autoria Própria

Para capturar o comando de ligar e desligar do ar-condicionado, foi desenvolvido um código, utilizando o receptor infravermelho conectado ao ESP32. A captura do RawData dos sinais infravermelhos permite que o código final seja mais enxuto, pois o microcontrolador possui uma memória limitada. Em vez de recapturar os sinais toda vez que o sistema é reiniciado, os comandos são armazenados previamente e inseridos diretamente no código. Dessa forma, o sistema pode reproduzir esses comandos sem necessidade de processamento adicional. Caso o sistema seja utilizado com outro modelo de ar-condicionado ou dispositivo, será necessário refazer a captura para adaptá-los ao novo equipamento, garantindo a compatibilidade e o correto funcionamento da automação.

Figura 26 – Código para captura do RAWDATA dos comandos IR

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

// Defina o pino do receptor IR
const uint16_t kRecvPin = 15;

// Inicializa o objeto IRrecv
IRrecv irrecv(kRecvPin);

// Variável para armazenar os dados decodificados
decode_results results;

void setup() {
  // Inicia a comunicação serial para exibir o RAW Data
  Serial.begin(115200);
  // Inicia o receptor IR
  irrecv.enableIRIn();
  Serial.println("Pronto para capturar sinais IR...");
}

void loop() {
  // Verifica se algum sinal foi recebido
  if (irrecv.decode(&results)) {
    // Exibe o RAW Data no Serial Monitor
    Serial.println(resultToSourceCode(&results));
    Serial.println();

    // Reinicia o receptor IR para capturar novos sinais
    irrecv.resume();
  }
```

Fonte: Autoria Própria

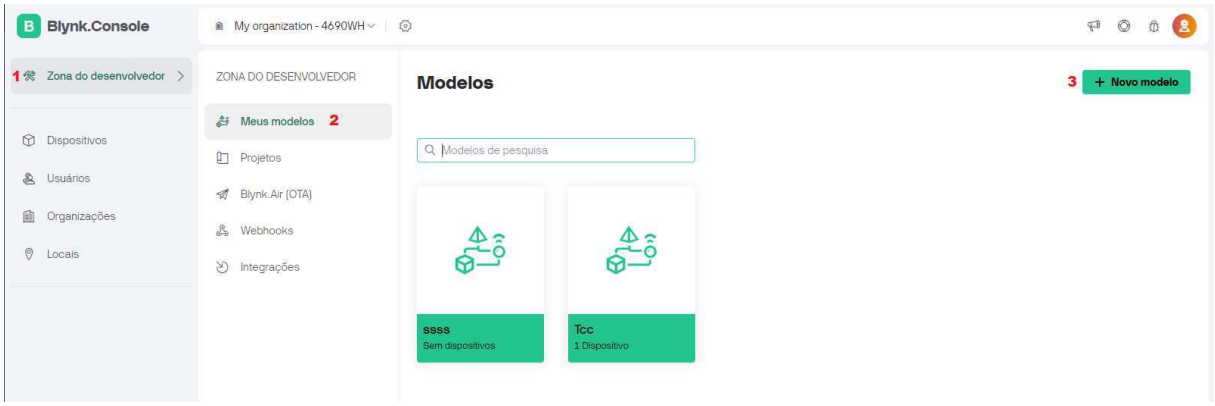
3.2.5 Conexão ESP32 e Blynk

A integração entre o ESP32 e o Blynk oferece uma solução eficiente para monitorar e controlar dispositivos remotamente via uma aplicação móvel. Para realizar essa conexão foi necessário realizar uma serie de passo que serão descritos a seguir:

1. Antes de conectar o ESP32 ao Blynk, foi necessario criar um conta na plataforma através do site oficial (https://blynk.io).

2. Após acessar com a conta logada, criamos um novo modelo indo em: Zona do desenvolvedor > Meus modelos > +Novo modelo.

Figura 27 – Representação do passo a passo para a criação de um novo modelo no Blynk



Fonte: Autoria Própria

3. Após dar um nome e especificar qual o microcontrolador será utilizado, seguimos para a configuração do modelo.

4. Na aba "Datastreams"foram adicionados Datasreams para cada componente presente no projeto. Eles são uma forma de estruturar os dados que entram e saem regularmente do dispositivo, onde cada dispositivo é atribuido a um pino virtual. Na Figura 28 são mostrados os Datastreams que foram adicionados ao modelo.

Figura 28 – Lista de Datastreams do projeto

Datastreams

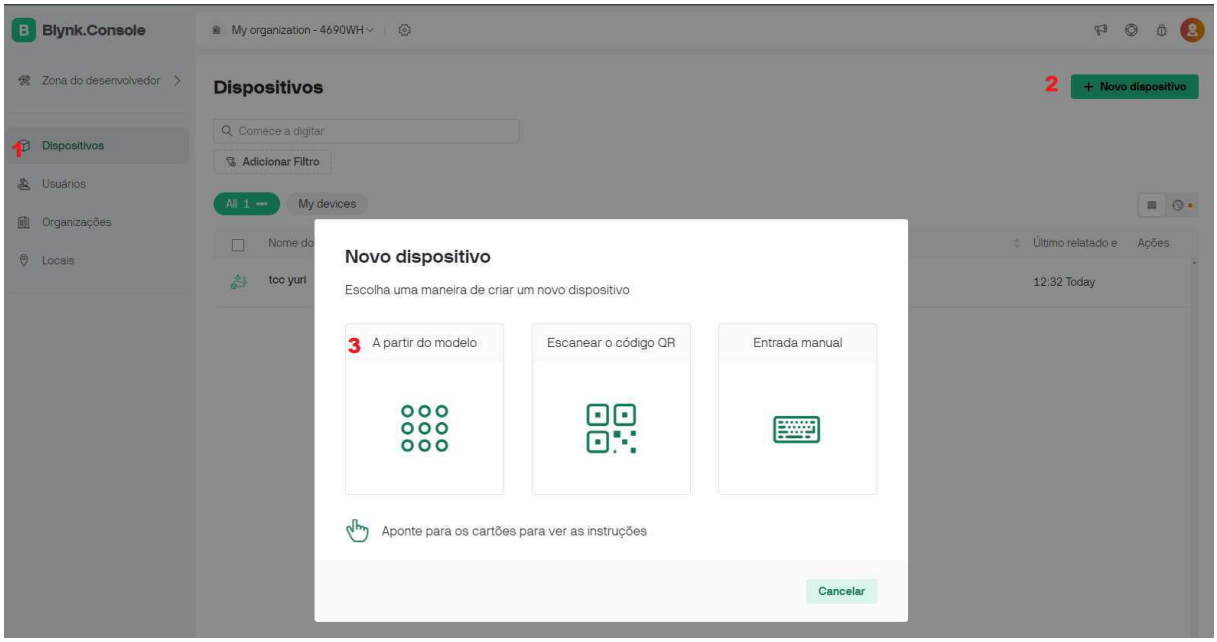
Pesquisar datastream

Id	Nome do usuário	Apelido	Cor	Pin	Tipo de ...	Unid...	É bruto	Mínimo	Máximo
1	Temperatura	Temperatura		V1	Inteiro	°C	false	0	100
2	Umidade	Umidade		V2	Inteiro	%	false	0	100
3	gases	gases		V3	String		false		
4	Alarme Temperatura	Alarme Temperatura		V4	Inteiro	°C	false	0	100
5	Alarme Umi	Alarme Umi		V0	Inteiro	%	false	0	100

Fonte: Autoria Própria

5. Após isso clicamos em: Dispositivos > + Novo dispositivo > A partir do modelo. Dessa forma criamos um dispositivo com todos os Datastreams que ja foram criados no modelo.

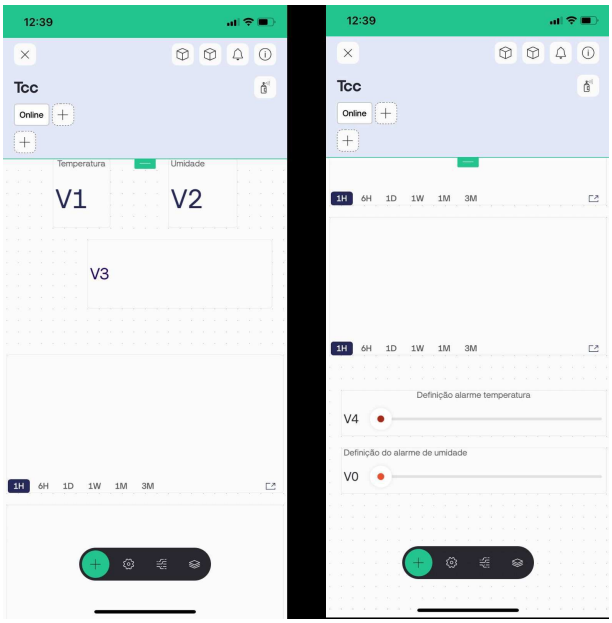
Figura 29 – Representação do passo a passo para a criação de um novo dispositivo no Blynk



Fonte: Autoria Própria

6. A partir dessa criação, o modelo pode ser acessado pelo aplicativo mobile do Blynk, e editado de forma que todos os dados pudessem ser visualizados da melhor forma como mostra na figura 30

Figura 30 – Tela do dispositivo com os widgets de monitoramento adicionados



Fonte: Autoria Própria

7. Com o Dispositivo criado, recebemos dados sobre o template, esses dados são essenciais para realização da comunicação entre ESP32 e Blynk, são eles: BLYNK_TEMPLATE_ID, BLYNK_TEMPLATE_NAME, BLYNK_AUTH_TOKEN.

8. configuraramos o código no ESP32 para se conectar ao Blynk utilizando um código base que é cedido pela plataforma do Blynk e modificado para nosso projeto especifico.

Figura 31 – Código base para conexão entre ESP32 e Blynk

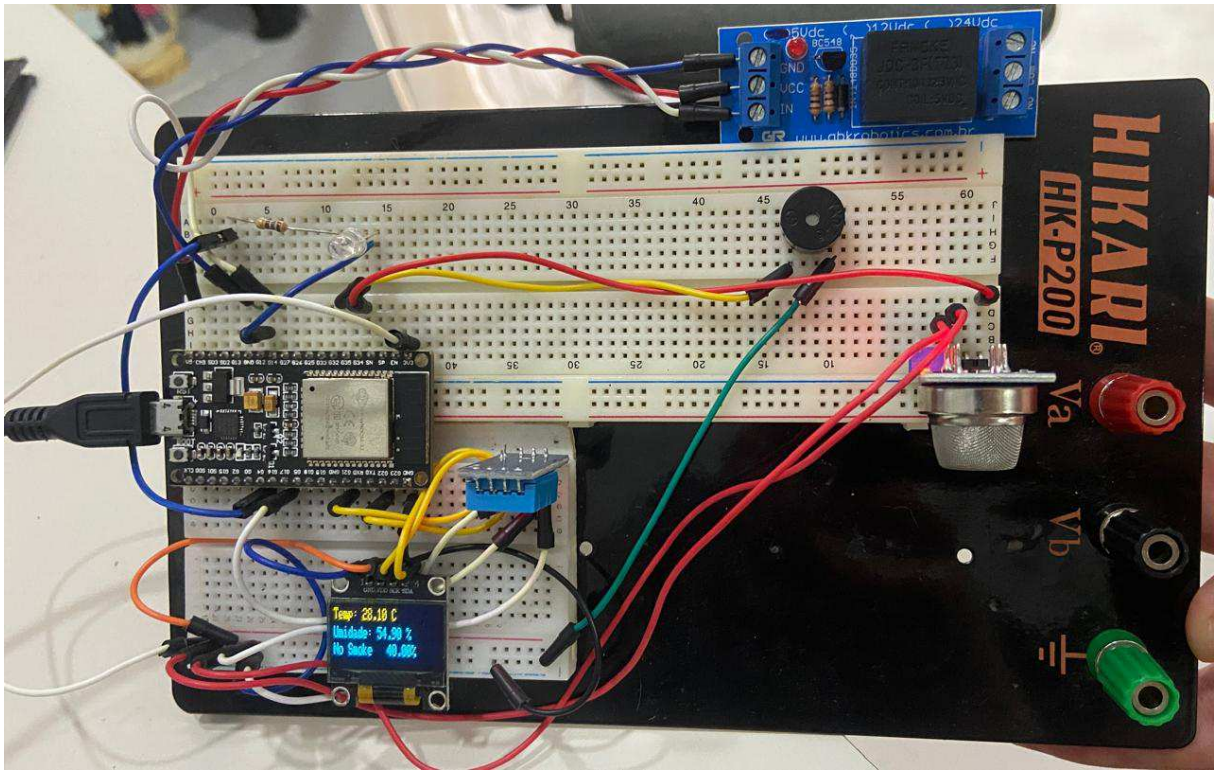
```
#define BLYNK_TEMPLATE_ID "Seu Template ID"
#define BLYNK_TEMPLATE_NAME "Nome do seu Projeto"
#define BLYNK_AUTH_TOKEN "Seu Token de Autenticação"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// Insira o nome da sua rede Wi-Fi e senha
char ssid[] = "Seu SSID";
char pass[] = "Sua Senha";

// Insira o token gerado pelo Blynk ao criar o projeto
char auth[] = BLYNK_AUTH_TOKEN;

void setup() {
  // Inicializa a comunicação serial
  Serial.begin(115200);

  // Conecta ao Blynk
  Blynk.begin(auth, ssid, pass);

  // Opcional: Mensagem indicando que o sistema está pronto
  Serial.println("Sistema conectado ao Blynk");
}

void loop() {
  // Roda o Blynk
  Blynk.run();
}
```

Fonte: Autoria Própria

3.2.6 Integração dos componentes

Após a realização de testes individuais com cada sensor e dispositivo, o processo de integração dos componentes foi iniciado. Nesse momento, todas as bibliotecas e módulos previamente testados, de forma separada, foram combinados para verificar se havia possíveis conflitos ou interferências entre eles. A integração pode ser vista na figura 32.

Figura 32 – Fotografia da montagem do projeto.



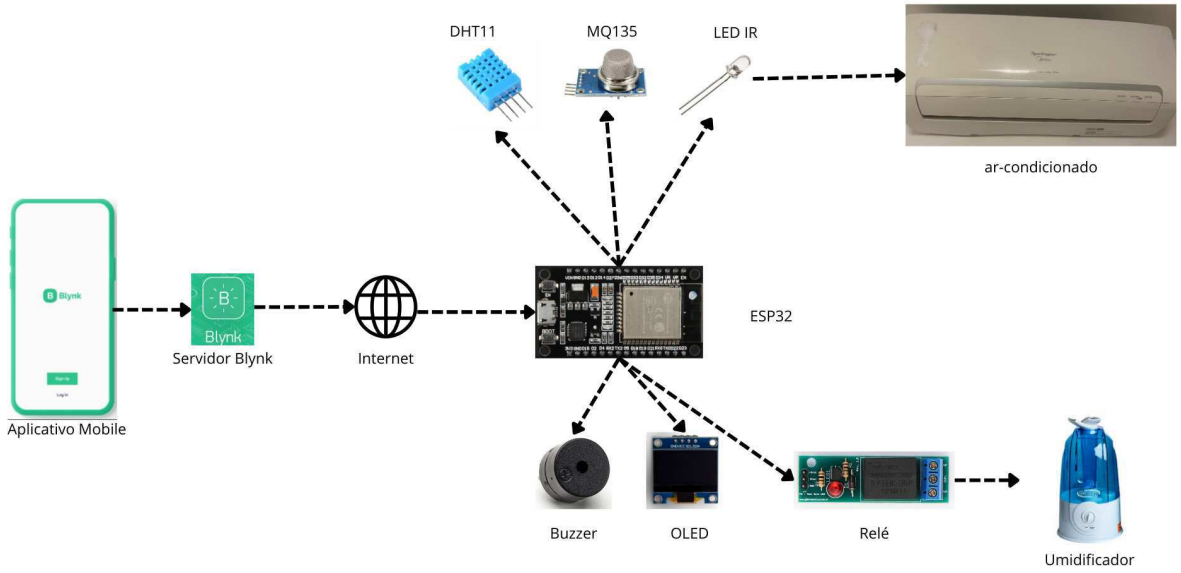
Fonte: Autoria Própria

Durante o teste de integração, o código do anexo A, foi carregado no ESP32 via USB, utilizando o ambiente de desenvolvimento Arduino IDE. Todos os componentes funcionaram de maneira integrada, com a comunicação sendo gerida tanto localmente, através do display OLED, quanto remotamente via o aplicativo Blynk IoT. Com a integração completa, o sistema permite não apenas a automação de dispositivos por controle infravermelho, mas também o monitoramento ambiental constante, com alertas e controle direto via smartphone.

4 Resultados Obtidos

Os resultados positivos obtidos durante os testes demonstraram que a integração dos sensores e dispositivos foi bem-sucedida, permitindo o prosseguimento do projeto sem complicações significativas. A implementação do sistema proposto foi concluída com sucesso, resultando em uma solução funcional capaz de monitorar as variáveis ambientais de um ambiente e controlar um umidificador e um ar-condicionado de maneira autônoma. A figura 33 mostra o diagrama com as conexões do projeto.

Figura 33 – Diagrama de conexões do Projeto



Fonte: Autoria Própria

O sensor DHT11 foi utilizado para capturar as variáveis de temperatura e umidade do ambiente. Esses dados são exibidos diretamente no display OLED, o que facilita o monitoramento local em tempo real. Além disso, no aplicativo Blynk, os dados coletados são organizados em display's e em gráficos de históricos, que permitem ao usuário visualizar as variações das variáveis ao longo do tempo. Esses gráficos podem ser ajustados para exibir informações de diferentes períodos, como 1 hora, 6 horas, 1 dia, 1 semana, 1 mês ou 3 meses, proporcionando uma análise detalhada do histórico ambiental.

Figura 34 – Tela Principal do sistema, demonstrando o valor atual das variáveis e seus gráficos de histórico

Fonte: Autoria Própria

O sistema também inclui slides de seleção no aplicativo móvel, permitindo ao usuário definir os níveis ideais de temperatura e umidade para o ambiente. Se esses limites definidos não forem atingidos, o sistema ajusta automaticamente os dispositivos conectados, como o ar-condicionado ou o umidificador, para garantir que o ambiente atinja as condições desejadas.

Figura 35 – Slider's para definição dos parâmetros desejados.

Fonte: Autoria Própria

Para facilitar o processamento dos dados coletados pelo sensor MQ135, que monitora a qualidade do ar, foi implementada uma conversão das leituras analógicas em uma porcentagem de 0 a 100%, utilizando a função map() e constrain() no código. Essa abordagem simplifica o tratamento dos dados e melhora a usabilidade das informações fornecidas pelo sensor, permitindo que o sistema identifique rapidamente variações na concentração de gases nocivos.

Figura 36 – Trecho do código que converte a leitura em um valor percentual entre 0 e 100.

```
// Converte o valor lido em uma porcentagem (0 a 100%)
float gasConcentrationPercentage = map(mq135Value, 0, ANALOG_MAX_VALUE, 0, 100);
gasConcentrationPercentage = constrain(gasConcentrationPercentage, 0, 100);
```

Fonte: Autoria Própria

Adicionalmente, foram configurados alertas no aplicativo Blynk para notificar o usuário quando a temperatura ou a umidade do ambiente excede os limites predefinidos.

Figura 37 – Notificações do aplicativo para as alterações das variáveis.

Fonte: Autoria Própria

Um ponto crítico do sistema é a detecção de gases perigosos, visto que, quando o sensor MQ135 identifica concentrações de gases acima de 45%, o nível de poluição já é considerado preocupante. Esse limite foi estabelecido com base em testes e na sensibilidade do sensor, uma vez que concentrações acima desse valor podem indicar a presença de substâncias potencialmente nocivas à saúde. Nesses casos, o sistema automaticamente liga o umidificador e, além de emitir alertas no aplicativo, também aciona um buzzer para alertar o usuário localmente. Utilizando a biblioteca Callmebot, o sistema envia notificações via

WhatsApp, garantindo que o usuário seja informado de diferentes formas e que uma ação corretiva possa ser tomada. Essas funcionalidades aumentam a segurança e a interatividade do sistema, oferecendo uma solução completa para o controle e monitoramento do ambiente.

Figura 38 – Alerta do aplicativo para a identificação de gases no ambiente.

Fonte: Autoria Própria

Figura 39 – Mensagens enviadas pelo CallmeBot via whatsapp.

Fonte: Autoria Própria

Uma limitação importante do projeto é que ele exige a captura prévia dos sinais de ligar e desligar do ar-condicionado utilizado. Isso significa que, para cada novo aparelho que o sistema precisará controlar, é necessário repetir o processo de captura de sinais infravermelhos, o que pode ser um entrave para a implementação em larga escala. Essa necessidade de captura manual torna o processo menos eficiente, especialmente em ambientes que possuem diferentes marcas e modelos de ar-condicionado, dificultando a automação de múltiplos dispositivos sem uma intervenção técnica.

Como sugestão para melhorias futuras, poderia ser implementado um banco de dados com sinais de controle de diferentes modelos e marcas de ar-condicionado. Esse banco de dados funcionaria como uma biblioteca centralizada que o sistema poderia acessar para identificar e reproduzir automaticamente os comandos IR corretos, sem a necessidade

de captura manual. Isso tornaria o sistema mais versátil, facilitando sua utilização em diversos cenários e ampliando sua aplicabilidade para ambientes comerciais e residenciais com equipamentos variados.

Com esses resultados, o sistema se mostrou eficiente em proporcionar um controle automatizado dos dispositivos conectados e monitorar de maneira confiável as condições ambientais, atingindo os objetivos de bem-estar e eficiência estabelecidos no projeto.

5 Considerações Finais

Esse trabalho teve como objetivo desenvolver um sistema de monitoramento de qualidade do ar utilizando um microcontrolador ESP32 integrado a um aplicativo móvel. Este sistema foi planejado para capturar dados de temperatura, umidade e concentração de gases, enviando essas informações ao aplicativo. Nele, o usuário pode definir níveis de alerta personalizados e configurar parâmetros para automatizar o ar-condicionado e o umidificador, além de acessar o histórico das medições, visando oferecer uma ferramenta prática e acessível, que permita ao usuário monitorar e controlar a qualidade do ar de forma contínua.

O desenvolvimento do sistema permitiu alcançar os objetivos propostos. A ESP32 mostrou-se eficiente na captura e transmissão dos dados, enquanto o Blynk possibilitou a interação amigável e remota com o usuário. A escolha dos sensores DHT11 e MQ135 contribuiu para o monitoramento efetivo das condições ambientais. Apesar dos desafios encontrados, como a necessidade de capturar os comandos de controle remoto de cada ar-condicionado utilizado, o projeto foi finalizado com sucesso, cumprindo as funcionalidades planejadas. A produção deste trabalho permitiu explorar diversas tecnologias e implementar um sistema de automação funcional que se mostrou viável para monitorar e melhorar a qualidade do ar de forma automatizada.

Para o futuro, sugere-se aprimorações ao projeto com a implementação de novas funcionalidades, visando torná-lo ainda mais específico e dedicado a necessidades distintas, como o monitoramento em ambientes para recém-nascidos. Seria possível integrar novos sensores, como de som e iluminação, e funcionalidades de controle de ventilação, para adequar o ambiente de forma mais personalizada. Além disso, o desenvolvimento de um banco de dados com comandos pré-configurados para diferentes modelos de ar-condicionado ampliaria a aplicação do sistema, tornando-o mais versátil e fácil de adaptar a outros cenários.

Referências

ADAFRUIT. *DHT11 Basic Temperature-Humidity Sensor*. 2020. Disponível em: <https://www.adafruit.com/product/386>. Citado na página 4.

Blynk. *Blynk Documentation*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://docs.blynk.io/en>. Citado na página 18.

BRUNEKREEF, B.; HOLGATE, S. Air pollution and health. *The Lancet*, Lancet Publishing Group, v. 360, n. 9341, p. 1233–1242, 2002. Disponível em: <https://dspace.library.uu.nl/handle/1874/7410>. Citado na página 4.

CETESB - Companhia Ambiental do Estado de São Paulo. *Relatório de Qualidade do Ar no Estado de São Paulo 2023*. 2023. Acesso em: 23 set. 2024. Disponível em: <https://cetesb.sp.gov.br/ar/wp-content/uploads/sites/28/2024/08/Relatorio-de-Qualidade-do-Ar-no-Estado-de-Sao-Paulo-2023.pdf>. Citado na página 3.

DOCS, H. Blynk mobile and web application. Acesso em: 25 de set. de 2024. 2024. Disponível em: <https://docs.hardwario.com/tower/platform-integrations/blynk-app/>. Citado na página 9.

ELECTRONICS, H. *MQ135 Gas Sensor Datasheet*. 2016. Disponível em: <https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-135.pdf>. Citado na página 5.

Eletrogate. *Buzzer Ativo 5V*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://www.eletrogate.com/buzzer-ativo-5v>. Citado na página 16.

Eletrogate. *Módulo Relé 1 Canal 5V*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://www.eletrogate.com/modulo-rele-1-canal-5v>. Citado na página 14.

FEIFEL, B. Qualidade do ar: Como manaus chegou a ter 3º pior ar do mundo. *Agência Pública*, 2023. Disponível em: <https://apublica.org/2023/11/queimada-como-manaus-chegou-a-estar-entre-as-3-piores-cidades-do-mundo-em-qualidade-do-ar/>. Citado na página 3.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Elsevier, v. 29, n. 7, p. 1645–1660, 2013. Citado na página 8.

KALAIVANI, K. et al. Smart energy conservation system with crime prevention and control. In: IEEE. *2023 Intelligent Computing and Control for Engineering and Business Systems (ICCEBS)*. [S.l.], 2023. p. 1–6. Citado na página 8.

LIBELIUM. *Waspmote Gases Sensor Guide*. 2017. Disponível em: <http://www.libelium.com>. Citado na página 5.

LIBELIUM. *Gases PRO Sensor Board*. [S.l.], 2020. Disponível em: <https://development.libelium.com/gases_pro_sensor_guide/gases-pro-sensor-board-calibrated>. Citado na página 11.

MAHETALIYA, S. et al. Iot based air quality index monitoring using esp32. *International Research Journal of Engineering and Technology*, v. 8, n. 4, p. 5186–5191, 2021. Citado na página 8.

MAKERHERO. Como funciona o sensor de gás mq-135? Acesso em: 23 de set. de 2024. 2022. Disponível em: <https://www.makerhero.com/blog/como-funciona-o-sensor-de-gas-mq-135/>. Citado na página 6.

MakerHero. *LED Emissor Infravermelho IR 5mm / Módulo Receptor Infravermelho IR KY-022*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://www.makerhero.com/produto/led-emissor-infravermelho-ir-5mm/>. Citado na página 15.

MakerHero. *Módulo Receptor Infravermelho IR KY-022*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://www.makerhero.com/produto/modulo-receptor-infravermelho-ir-ky-022/>. Citado na página 15.

MAKERHERO. Sensor de umidade e temperatura dht11. Acesso em: 23 de set. de 2024. 2023. Disponível em: <https://www.makerhero.com/produto/sensor-de-umidade-e-temperatura-dht11/>. Citado na página 5.

MARKER, C. do. Esp32 pinout: Detalhes e conexões. Acesso em: 25 de set. de 2024. 2024. Disponível em: <https://clubedomaker.com/esp32-pinout>. Citado na página 7.

MAYNARD, R. L. et al. *Air pollution and health*. [S.l.]: Elsevier, 1999. Citado na página 1.

MONK, S.; MCCABE, M. *Programming Arduino: getting started with sketches*. [S.l.]: McGraw-Hill Education New York, 2016. v. 176. Citado 4 vezes nas páginas 1, 4, 7 e 10.

POPE, C.; DOCKERY, D. Health effects of fine particulate air pollution: Lines that connect. *Journal of the Air & Waste Management Association*, Taylor & Francis, v. 56, n. 6, p. 709–742, 2006. Disponível em: <https://core.ac.uk/display/1449268>. Citado na página 4.

RoboCore. *Display OLED 0.96"I2C Branco*. 2023. Acessado em: 05 out. 2023. Disponível em: <https://www.robocore.net/display/display-oled-96-i2c-branco>. Citado na página 16.

SPENGLER, J. D.; SAMET, J. M.; MCCARTHY, J. F. *Indoor air quality handbook*. [S.l.]: McGraw-Hill Professional, 2000. Citado na página 1.

SYSTEMS, E. *ESP32 Series Datasheet*. [S.l.], 2019. Disponível em: <https://www.espressif.com/en/products/socs/esp32/resources>. Citado 2 vezes nas páginas 8 e 9.

World Health Organization. *Air Quality Guidelines: Global Update 2021*. World Health Organization, 2021. Disponível em: <https://www.who.int/publications/i/item/9789240034228>. Citado na página 3.

ANEXO A – Código Implementado

Neste apêndice é apresentado o código para a implementação do projeto realizado.

```
#define BLYNK_TEMPLATE_ID "TMPL2fb2CyhuL"
#define BLYNK_TEMPLATE_NAME "Tcc"
#define BLYNK_AUTH_TOKEN "xcmFUAaRbn3nunk_bE8LcKqN24HerWmp"

#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRsend.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <DHT.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Callmebot_ESP32.h>

#include <ESP.h>

// Configura  es do display OLED
#define SCREEN_WIDTH 128 // Largura do display OLED em pixels
#define SCREEN_HEIGHT 32 // Altura do display OLED em pixels
#define OLED_RESET -1     // Reset pin # (ou -1 se compartilhar o pino de
    reset do Arduino)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Configura  es do DHT11
#define DHTPIN 17       // Pino onde o DHT11 est  conectado (GPIO 17)
#define DHTTYPE DHT11 // Tipo do sensor DHT
DHT dht(DHTPIN, DHTTYPE);

// Configura  es do MQ135
#define MQ135PIN 34 // Pino onde o MQ135 est  conectado (GPIO 34)

// Configura  es para leitura anal gica do MQ135
#define ANALOG_MAX_VALUE 4095 // Valor m ximo da leitura anal gica (para
    ESP32, 12 bits = 2^12 - 1)
#define MAX_GAS_CONCENTRATION 1000 // Valor m ximo para gases (ajuste
    conforme necess rio)

// Token de Autentica  o do Blynk
char auth[] = "xcmFUAaRbn3nunk_bE8LcKqN24HerWmp";

// Configura  es da Rede Wi-Fi
```

```
39 char ssid[] = "RE085_2.4G_5652B4";
40 char pass[] = "yuri1234";
41
42 // Vari vel para armazenar a temperatura de alarme
43 float alarmTemperature = 40; // Valor padr o
44
45 BLYNK_WRITE(V4) { // Esta fun  o  chamada sempre que o valor do Slider
       alterado
46   alarmTemperature = param.asFloat(); // Define a temperatura de alarme com
       o valor do Slider
47 }
48
49 // Vari vel para armazenar a Umidade de alarme
50 float alarmUmid = 40; // Valor padr o
51
52 BLYNK_WRITE(V0) { // Esta fun  o  chamada sempre que o valor do Slider
       alterado
53   alarmUmid = param.asFloat(); // Define a temperatura de alarme com o
       valor do Slider
54 }
55
56 String phoneNumber = "+558382132540";
57 String apiKey = "4040042";
58
59
60 int blockArl = 0;
61 int blockArd = 1;
62 int blockSmoke = 0;
63 int blockUml = 0;
64 int blockUmd = 1;
65
66 const int relayPin = 14;// Pino conectado ao rel
67
68 const uint16_t irSendPin = 4;  // Pino do emissor infravermelho
69 IRsend irsend(irSendPin);
70 const int buzzerPin = 32;
71
72
73 void setup() {
74   // Inicializa o Serial Monitor
75
76   Serial.begin(115200);
77   irsend.begin();  // Inicializa o emissor IR
78   pinMode(buzzerPin, OUTPUT); // Define o pino do buzzer como sa da
79   digitalWrite(buzzerPin, LOW); // Inicialmente o buzzer est  desligado
80   pinMode(relayPin, OUTPUT); // Define o pino do rel  como sa da
81   digitalWrite(relayPin, LOW); // Desliga o rel  inicialmente (
```

```
    naturalmente aberto)
82
83
84  WiFi.begin(ssid, pass);
85  Serial.println("Connecting");
86  while(WiFi.status() != WL_CONNECTED) {
87    delay(500);
88    Serial.print(".");
89  }
90  Serial.println("");
91  Serial.print("Connected to WiFi network with IP Address: ");
92  Serial.println(WiFi.localIP());
93  Serial.print("Connected");
94
95
96
97  // Inicializa a conex o com o Blynk
98  Blynk.begin(auth, ssid, pass);
99
100 // Inicializa o display OLED
101 if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Endere o padr o
    para displays OLED de 128x32
102   Serial.println(F("Falha na inicializa   o do OLED"));
103   for (;;);
104 }
105
106 // Inicializa o DHT11
107 dht.begin();
108
109 // Mensagem inicial no OLED
110 display.clearDisplay();
111 display.setTextSize(1);
112 display.setTextColor(WHITE);
113 display.setCursor(0, 0);
114 display.println("Inicializando...");
115 display.display();
116 delay(2000); // Aguarda 2 segundos para visualiza   o
117 }
118
119 void loop() {
120 // Atualiza o Blynk
121 Blynk.run();
122
123 // Leitura da umidade e temperatura do sensor DHT11
124 float humidity = dht.readHumidity();
125 float temperature = dht.readTemperature();
126
```

```
127   // Leitura do MQ135
128   int mq135Value = analogRead(MQ135PIN);
129
130   // Converte o valor lido em uma porcentagem (0 a 100%)
131   float gasConcentrationPercentage = map(mq135Value, 0, ANALOG_MAX_VALUE, 0
        , 100);
132   gasConcentrationPercentage = constrain(gasConcentrationPercentage, 0, 100
        );
133
134   // Limpa o display OLED
135   display.clearDisplay();
136
137   // Exibe temperatura no OLED
138   display.setTextSize(1);                  // Tamanho do texto
139   display.setTextColor(WHITE);             // Cor do texto
140   display.setCursor(0, 0);                 // Posi   o do texto
141   display.print("Temp: ");
142   display.print(temperature);
143   display.println(" C");
144
145   // Exibe umidade no OLED
146   display.setCursor(0, 10); // Ajusta posi   o para segunda linha
147   display.print("Umidade: ");
148   display.print(humidity);
149   display.println(" %");
150
151   // Determina a mensagem de acordo com a concentra   o de g s
152   String gasMessage;
153   String gasMessageBlynk;
154   if (gasConcentrationPercentage < 99) {
155     gasMessage = "No Smoke    ";
156     gasMessageBlynk = "Sem Fuma a";
157
158   } else {
159      digitalWrite(buzzerPin, HIGH); // Liga o buzzer se houver fuma a
160     gasMessage = "Smoke Detected ";
161     gasMessageBlynk = "Fuma a detectada";
162     Blynk.logEvent("smoke_detected", "O Umidificador de ar foi ligado, mas,
          recomendado que verifique o ambiente.");
163     if(blockSmoke == 0){
164       Callmebot.whatsappMessage(phoneNumber, apiKey, "*FUMA A DETECTADA*")
        ;
165       blockSmoke = 1;
166       digitalWrite(relayPin, HIGH); //aqui vem a logica deligar o rel
167     }
168
169   }
```

```
170   if (gasConcentrationPercentage < 23) {
171     digitalWrite(buzzerPin, LOW); // Desliga o buzzer se n o houver
      fuma a
172     if (blockSmoke == 1){
173       digitalWrite(relayPin, LOW);//aqui vem a logica dedesligar rel
174       blockSmoke=0;
175     }
176   }
177
178    // Determina a mensagem de acordo com a o Slider de Temperatura
179   if (temperature > alarmTemperature) {
180     blockArd =0;
181     if (blockArl == 0){
182       Blynk.logEvent("alarm_temperature", "Seu Ar-condicionado foi ligado")
      ;
183       blockArl = 1;
184       uint64_t mideaCode1 = 0xD962CD58;
185       Serial.println("Enviando c digo Midea 0xD962CD58...");
186       irsend.sendMidea(mideaCode1, 32);  // Envia o c digo Midea com 32
      bits
187       //aqui falta a l gica do IRremote
188     }
189
190   }
191   if ((temperature + 3) < alarmTemperature) {
192       blockArl = 0;
193       if (blockArd == 0){
194         Blynk.logEvent("temperatura_estabilizada", "Seu Ar-condicionado foi
       desligado");
195         blockArd =1;
196         uint64_t mideaCode2 = 0x5E7E4778;
197           Serial.println("Enviando c digo Midea 0x5E7E4778...");
198       irsend.sendMidea(mideaCode2, 32);  // Envia o segundo c digo Midea
      com 32 bits;
199         //aqui falta a l gica do IRremote
200       }
201
202
203   }
204
205    // Determina a mensagem de acordo com a o Slider de Temperatura
206   if (humidity < alarmUmid) {
207     blockUmd =0;
208     if (blockUml ==0){
209       Blynk.logEvent("alarm_umi", "Seu Umidificador foi ligado");
210       blockUml =1;
211       digitalWrite(relayPin, HIGH); //ligando o rel
```

```
212     }
213
214   }
215   if ((humidity-5) > alarmUmid) {
216     blockUml =0;
217     if(blockUmd == 0){
218       Blynk.logEvent("umidade_estabilizada", "Seu Umidificador foi
      desligado");
219       blockUmd = 1;
220       digitalWrite(relayPin, LOW); //aqui vem a logica deligar o rel
221     }
222
223   }
224   // Exibe mensagem no OLED
225   display.setCursor(0, 20); // Ajusta posi  o para terceira linha
226   display.print(gasMessage);
227   display.print(gasConcentrationPercentage);
228   display.print("%");
229
230   // Atualiza o display com as novas informa  es
231   display.display();
232
233   // Envia os dados para o Blynk
234   Blynk.virtualWrite(V1, temperature);          // Envia temperatura para o
       display V1
235   Blynk.virtualWrite(V2, humidity);             // Envia umidade para o
      display V2
236   Blynk.virtualWrite(V3, gasMessageBlynk);           // Envia a mensagem de
       g s para o display V3
237
238   // Delay para pr xima leitura (2 segundos)
239   delay(2000);
240 }
```

Codigo/TCCpronto.ino