Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Desenvolvimento de um sistema para o monitoramento, automação e controle de ar-condicionados

Igor Alves de Souza

Campina Grande - PB
Novembro de 2023

Igor Alves de Souza

Desenvolvimento de um sistema para o monitoramento, automação e controle de ar-condicionados

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletricista.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Coordenação de Graduação em Engenharia Elétrica - CGEE

Jaidilson Jó da Silva, D.Sc.

(Orientador)

Campina Grande - PB

Novembro de 2023

Igor Alves de Souza

Desenvolvimento de um sistema para o monitoramento, automação e controle de ar-condicionados

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletricista.

Aprovado em ____ /____ /____

George Acioli Júnior
Universidade Federal de Campina Grande
Avaliador

Jaidilson Jó da Silva
Universidade Federal de Campina Grande
Orientador

Campina Grande - PB
Novembro de 2023

Dedico este trabalho a minha família e amigos, por serem o motivo de chegar onde cheguei.

Agradecimentos

Dedico esse trabalho a toda minha família, que sempre torceram por mim, e estarão sempre torcendo.

Gostaria de agradecer ao meu professor e orientador Jaidilson Jó da Silva, com quem aprendi diversas técnicas no decorrer de suas disciplinas, que esteve sempre presente disponível no desenvolvimento deste trabalho, e por ser uma grande referência para mim durante a graduação.

Também quero agradecer a todos os meus amigos que fiz durante a graduação, por tornarem essa trajetória mais divertida e por sempre poder contar com eles.

"The most important step a man can take.
It's not the first one, is it? It's the next one. Always the next step",
(Brandon Sanderson)

Resumo

Dispositivos inteligentes que podem ser controlados apenas com o uso da internet vem sendo cada vez comum no ambiente residencial e na industria. Com o avanço da industria 4.0 e da IoT, dispositivos conectados na rede estão cada vez mais comuns. E um desses dispositivos aplamente utilizado que estão cada vez mais conectados são os condicionadores de ar. Neste trabalho foi desenvolvido um sistema para monitoramento e controle de ar-condiciandos por meio da conexão da internet, utilizando uma placa de desenvolvimento ESP32. Além disso foi utilizada ferramentas como plataformas IoT que permite o gerenciamento remoto de dispositivos, e integração do projeto com auto-falante inteligente.

Palavras-chave: Automação, IoT, Indústria 4.0, Dispositivos Inteligentes, ESP32, Assistentes virtuais.

Abstract

Smart devices that can be controlled just using the internet are becoming increasingly common in residential and industrial environments. With the advancement of industry 4.0 and IoT, devices connected to the network are increasingly common. And one of these widely used devices that are increasingly connected are air conditioners. In this work, a system was developed for monitoring and controlling air conditioning through an internet connection, using an ESP32 development board. In addition, tools such as IoT platforms were used, which allows remote management of devices, and integration of the project with smart speakers.

Keywords: Automation, IoT, Industry 4.0, Smart Devices, ESP32, Virtual Assistants.

Lista de ilustrações

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
UFCG	Universidade Federal de Campina Grande
IoT	*Internet of Things*
LED	*Light Emitting Diode*
IR	*Infrared*
PDM	*Pulse Distance Modulation*
AI	*Artificial Intelligence*
NLU	*Natural Language Understanding*
IDE	*Integrated Development Environment*

Sumário

1 Introdução

A energia elétrica é um recurso essencial com demanda em constante crescimento. De acordo com o Relatório Síntese do Balanço Energético Nacional, elaborado e publicado anualmente pela EPE, Empresa de Pesquisa Energética, o consumo final de eletricidade no país em 2022 cresceu 2,3%. Os setores que mais contribuíram para este avanço foram o Comercial, seguido pelo Industrial e Residencial. O consumo de energia elétrica está crescendo cada vez mais, com diversos aparelhos eletrônicos sendo ligados diariamente, sendo alguns desses utilizados 24 horas por dia (EPE, 2023).

De acordo com a nota técnica 030/2018 da Empresa de Pesquisa Energética, a energia utilizada para o conforto térmico é o uso final em edifícios que mais cresce no mundo. Já no Brasil, apenas no setor residencial, é estimado que o uso de ar-condicionado pelas famílias tenha mais do que duplicado entre 2005 e 2017. O aumento da quantidade de unidades vendidas permitiu que os condicionadores de ar apresentassem um ganho relevante na participação do consumo total de eletricidade nos últimos anos, cuja participação passou de 7% em 2005 para 14% em 2017 (EPE, 2018).

Visando a diminuição de um desperdício de forma econômica, fica claro a necessidade de soluções de otimização com sistemas que possam monitorar e controlar ambientes de modo que seja não só possível reduzir os gastos com a energia elétrica, mas também proporcionar um maior conforto aos usuários.

O mercado da Internet das Coisas (IoT) cresceu rapidamente nos últimos anos após o aumento na procura de comunicação e controle para múltiplos dispositivos, se tornando umas das tecnologias mais importantes atualmente (MAIER; SHARP; VAGAPOV, 2017). A Internet das Coisa descreve a rede de objetos físicos, softwares e outras tecnologias integradas a sensores com o objetivo de conectar e compartilhar dados com dispositivos e sistemas pela internet. Com ela tivemos capacidades mais avançados de comunicação que mudaram drasticamente as propriedades e a operação da automação industrial e da manufatura, gerenciamento de negócios e de processos, transporte e da logística inteligentes (ATZORI; IERA; MORABITO, 2010).

Umas das principais vantagens é a redução de custos e uma maior flexibilidade ao reduzir processos manuais e automatizar tarefas repetitivas (KHANCHUEA; SIRIPOKARPIROM, 2019). Por exemplo, os dispositivos IoT podem ser utilizados para monitorizar o uso de energia e otimizar o consumo. Com esse crescimento veio também a popularização de sistemas embarcados microcontrolados, como por exemplo o Arduino, Raspberry Pi, e o ESP32, tornando possível a criação de projetos de automação para atividades cotidianas de forma fácil.

1.1 Objetivos

Estudar e desenvolver um dispositivo IoT utilizando microcontroladores com o objetivo de automatizar o controle de ar-condicionado, com o propósito de garantir a compatibilidade do sistema para qualquer tipo de ambiente, diminuindo assim o consumo de energia. A proposta visa explorar recursos e técnicas para a criação de um dispositivo de fácil uso, e desenvolver um algoritmo que garante a eficiência no monitoramento e controle de ar-condicionado.

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 apresentada uma revisão da literatura sobre os princípios de funcionnamento de ar-condicionados, junto das diversas variações que existem. Também será apresentado o funcionamento dos diodos emissores de luz infravermelha, que é um dos métodos de comunicação utilizados nos controles de condicionadores de ar, e também nos diversos formatos de protocolos e suas aplicações. E por fim será introduzido os assistentes de voz utilizados nos dias de hoje.

2.1 Ar-condicionados

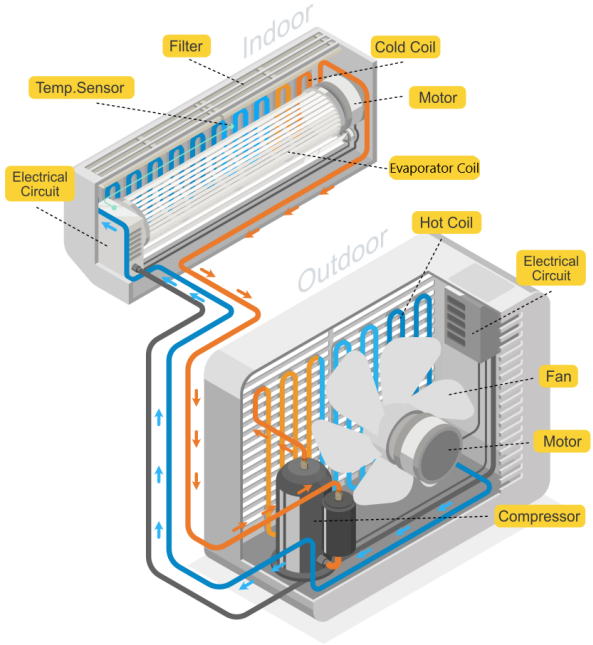
O condicionamento de ar é o processo de resfriamento e desumidificação do ar interno para atender aos requisitos de conforto térmico ou outros fins. Com o sistema de condicionamento de ar é possível realizar o controle da temperatura, umidade e qualidade do ar, com funções de aquecimento/resfriamento, umidificação e desumidificação (HALL, 2010).

O ar-condicionado funciona com base nos princípios da mudança de fase, que é a transformação de um material de um estado (ou fase) da matéria para outro, como exemplo um material que muda da forma líquida para gasosa. Quando ocorre uma mudança de líquido para gás, o material absorve calor. Por outro lado, quando o material muda de gás para líquido, ele libera calor. Um ar-condicionado é uma máquina que força a mudança de fase utilizando os princípios de transferência de calor resultantes para resfriar ambientes.

Os condicionadores de ar são compostos de muitos componentes, mas independente do tipo de ar-condicionado, todos terão os principais: um Compressor, Condensador, Válvula de Expansão e um Evaporador, como mostrado na Figura 1. Também é utilizado um gás refrigerante específico dentro da tubulação que percorre o sistema, para passar pelo processo de mudança de fase (ARCHTOOLBOX, 2021).

O compressor é o coração do ar-condicionado, e o que mais consome energia no sistema. Ele aumenta a densidade do gás refrigerante, fazendo que a pressão e temperatura aumente. Esse gás quente viaja então para o condensador, onde se move através de uma série de bobinas com finas lâminas de metal fixadas, chamada de serpentina. Essa passagem pelo condensador faz com que o gás libere calor e mude de fase de gás para líquido. O calor gerado pela mudança de estado é então soprado para o ambiente externo com a ajuda de ventiladores (DAIKIN, 2022).

Figura 1 – Ilustração de componentes de um Ar-condicionado.



Fonte: Adaptado de (ENGINEERING, 2023)

Então o refrigerante em estado líquido se expande em gás após passar pela válvula de expansão, que como o nome sugere, permite que o refrigerante comprimido se expanda. Quando isso acontece, o refrigerante sofre uma queda na pressão e na temperatura (DAIKIN, 2022). Esse refrigerante gasoso e resfriado vai para as serpentinas do evaporador para absorver o calor do ambiente. Com um ventilador, o ar é empurrado pelas serpentinas frias, o que retira o calor do ar, fazendo com que o ar esfrie. A finalidade do compressor no ciclo de compressão de vapor é comprimir o gás seco de baixa pressão do evaporador e aumentar sua pressão até a do condensador (HUNDY; TROTT; WELCH, 2016). A transferência de calor para o refrigerante faz com que ele volte a ser um vapor quente, iniciando o ciclo de refrigeração.

Outros componentes podem ser encontrados, como os filtros de ar com a função de manter a sujeira, poeira e outras partículas transportadas pelo ar longe do ar-condicionado. E também um termostato, que é responsável por manter a temperatura desejada.

2.2 Diodo emissor de infravermelho

Para podermos entender como é dado o controle de um ar-condicionado, primeiramente devemos saber como é o circuito do controle remoto. O controle remoto possui um microcontrolador, onde fica armazenado todas as informações de códigos referente ao aparelho, componentes eletrônicos e um diodo emissor de infravermelho, responsável por enviar as informações de controle.

A comunicação infravermelha está entre os métodos de comunicação sem fio mais simples e serve como uma forma econômica de transmitir bits de dados sem fio. Com infravermelho também temos a vantagem de uma alta relação sinal-ruído, possuir uma transmissão de informações confiável, forte contra interferências, e possuir baixo consumo de energia, sendo amplamente utilizado em eletrodomésticos. Além disso, também é utilizado na área de controle industrial, aeroespacial, segurança e muito mais (LV et al., 2010).

Os LEDs infravermelhos (IR) produzem uma luz que não é visível ao olho humano. Normalmente, o comprimento de onda da luz emitida por esses dispositivos está entre 800 e 980 nanômetros (KITABAYASHI et al., 2010). No entanto, os LEDs infravermelhos não são a única coisa que pode emitir ondas infravermelhas ou quase infravermelhas. Muitas outras fontes, como lâmpadas e o próprio sol, emitem ondas infravermelhas, o que é uma das dificuldades quando se trata de comunicações infravermelhas.

Além disso, qualquer pessoa pode enviar sinais IR. Uma TV, por exemplo, pode ser controlada por qualquer controle remoto que trabalhe com o mesmo protocolo, que nada mais é do que um acordo entre o remetente e o destinatário dos dados. Onde ambas as partes concordam em seguir um padrão predefinido e transmitir as informações de uma determinada forma. O método mais simples para transmitir valores binários com um LED IR seria ligar o LED infravermelho, enviando um 1 lógico, ou manter desligado, que representaria um 0 lógico, por um determinado período. Infelizmente, muitas outras fontes emitem radiação infravermelha, e o receptor não seria capaz de filtrar sinais indesejados de outras fontes.

Para superar esse problema, o emissor é obrigado a ligar e desligar o LED muito rapidamente, em vez de apenas ligá-lo e desligá-lo, normalmente, usando uma frequência de 38.000 Hz, que também é chamada de frequência portadora do sinal IR.

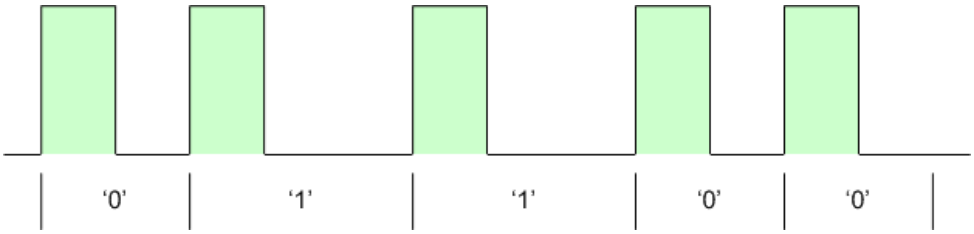
Porém, cada empresa utiliza diferentes tipos de protocolos de comunicação para enviar os sinais. A consequência disso é que cada dispositivo deve ser adequado para cada aplicação. Os dispositivos de controle remoto utilizados são compostos de chips de codificação fornecidos geralmente pela PHILIPS, NEC, Toshiba e outros, onde cada uma possui seu próprio protocolo para transmissão de sinais IR (LV et al., 2010). Os protocolos descrevem como os bits de dados devem ser dispostos para que ambas as partes possam compreender o que significam.

2.2.1 Protocolo NEC

O protocolo NEC é desenvolvido pela NEC, atual *Renesas*, e utiliza a chamada *Pulse Code Modulation* (Modulação por Código de Pulsos). Neste método, os dados são codificados modulando a duração do espaço entre os pulsos, enquanto o tempo do pulso

permanece constante, como mostrado na figura 2. O bit lógico '1' leva 2,25ms para ser transmitido, enquanto o tempo do bit lógico '0' leva 1,12ms, apenas metade da lógica '1' (LV et al., 2010). Os pulsos duram cerca de 562,5 us no início da lógica '0' e '1', com uma uma frequência portadora de 38 kHz (26,3 µs).

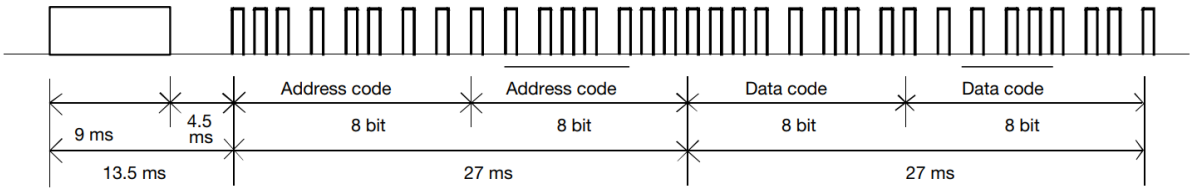
Figura 2 – Pulse Distance Encoding.



Fonte: (ALTIUM, 2017a).

No protocolo NEC, a mensagem transmitida consiste em uma rajada inicial 9ms seguido de um espaço de 4.5ms. O objetivo original deste LEAD CODE era permitir que as malhas de controle interno nos módulos receptores se estabilizassem (SEMICONDUCTORS, 2019). Após o LEAD CODE, o próximo trem de pulsos consiste em 8 bits de endereço usado para identificar o dispositivo a ser controlado, e outros 8 bits são usados para a transmissão dos dados de comando, que são transmitidos duas vezes. Na segunda vez todos os bits são invertidos e podem ser usados para verificação da mensagem recebida. Detalhes do protocolo NEC são mostrados na figura 3 abaixo.

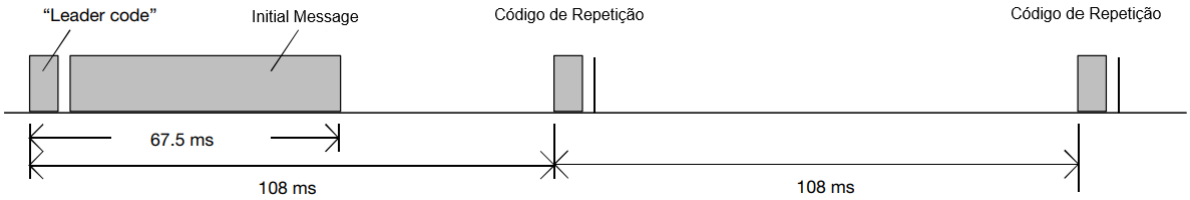
Figura 3 – Códigos de transmissão NEC.



Fonte: Vishay Semiconductors, 2019.

Se o botão do controle remoto for mantido pressionado, um código de repetição será emitido, normalmente 40 ms após a mensagem inicial terminar (ALTIUM, 2017b). O código de repetição é simplesmente um pulso de 9 ms do LEAD CODE, seguido por um espaço de 2,25 ms e um único bit para marcar o fim do código de repetição, que continuará a ser enviado em intervalos de 108 ms até que o botão seja solto.

Figura 4 – Exemplo de código de repetição.

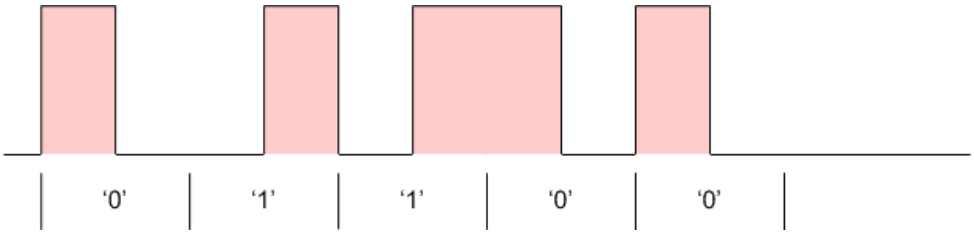


Fonte: Adaptado de Vishay Semiconductors, 2019.

2.2.2 Protocolo RC-5

O protocolo Philips RC-5 é desenvolvido pela Philips, e tem sido amplamente utilizado na produção eletrônica (LV et al., 2010). Ultimamente a Philips começou a usar um novo protocolo chamado RC-6 que possui mais recursos. Ao contrário do protocolo NEC, a Philips decidiu utilizar uma codificação chamada *Manchester* para distinguir entre os bits lógicos nas mensagens transmitidas. Neste método de codificação, todos os bits têm o mesmo comprimento, sendo metade do período de bits um pulso e a outra metade um espaço, com uma frequência portadora de 36kHz (27,7us).

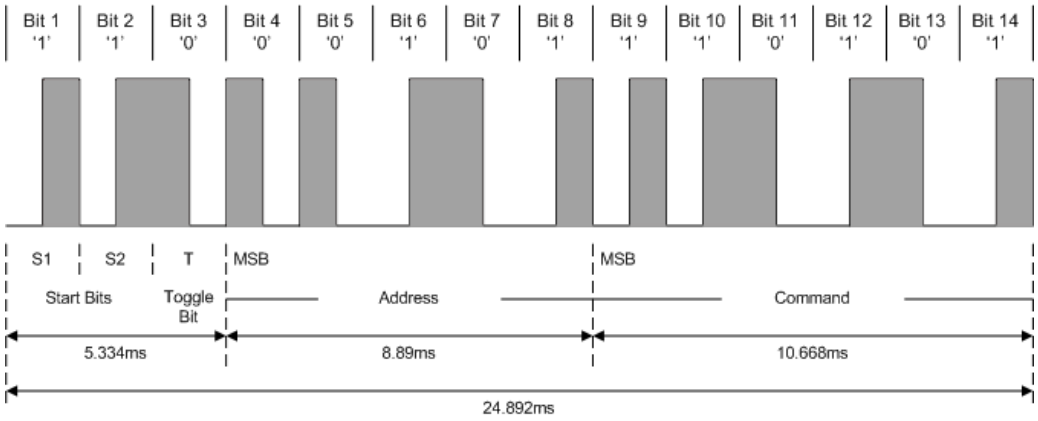
Figura 5 – Exemplo de codificação *Manchester*



Fonte: (ALTIUM, 2017a).

No protocolo RC-5, mostrado na figura 6, os dois primeiros pulsos são os *Start Bits*, ambos são "1"lógico, seguido de um *Toggle Bit*, que muda de valor a cada novo pressionar de botão (SEMICONDUCTORS, 2019). Desta forma, o receptor pode distinguir entre uma tecla que permanece pressionada ou que é pressionada repetidamente. Os próximos cinco bits representam o endereço do dispositivo a ser controlado. O endereço é seguido por 6 bits de comando que contêm as informações a serem transmitidas (LV et al., 2010). Enquanto um botão permanecer pressionado, a mensagem será repetida a cada 114ms. O *Toggle Bit* manterá o mesmo nível lógico durante todas essas mensagens repetidas.

Figura 6 – Códigos de transmissão RC-5



Fonte: (ALTIUM, 2017b).

Os métodos de codificação são considerações básicas para qualquer transmissão infravermelha. No entanto, o formato real da mensagem transmitida varia entre os fabricantes, podendo por exemplo, ter diferentes números de bits de endereço e de comando, pulsos adicionais antes e/ou depois dos bits de endereço e de comando, verificação de erros integrada e assim por diante (ALTIUM, 2017a).

2.3 Assistentes Virtuais

Assistente Virtual (também conhecido como "AI Assistant") é um aplicativo que pode compreender comandos de voz em linguagem natural e realizar tarefas para os usuários (ISLAM et al., 2022), que possibilita uma série de recursos que facilitam o dia a dia de quem o utiliza. Assistentes virtuais ganharam enorme popularidade no mundo contemporâneo. De acordo com a *Statista*, existem 4,2 bilhões de assistentes de voz em uso em 2023. É estimado que esse número chegará a 8,4 bilhões em 2024 (LARICCHIA, 2022).

As funcionalidades que os assistentes virtuais, também chamados de assistente de voz, pode proporcionar estão cada vez maiores e sua versatilidade aumenta com a integração a outros dispositivos, que podem ser ligados ou até programados com a voz do usuário, o que permite criar as chamadas casas inteligentes. Dispositivos integrados e conectados ao assistente de voz garantem autonomia e agilidade para realizar diversas tarefas, oferecendo maior comodidade dentro de casa.

Os assistentes virtuais são normalmente programas baseados em nuvem que requerem dispositivos e aplicativos conectados à Internet para funcionar. A maioria dos assistentes virtuais de Inteligência Artificial pode ser encontrado em dispositivos como smartphones, alto-falantes inteligentes ou outras plataformas, incluindo até aplicativos de mensagens (YASAR; BOTELHO, 2023).

Alguns dos assistentes de voz mais conhecidos incluem:

- **Siri.** O assistente virtual pessoal integrado e controlado por voz da Apple está disponível em dispositivos que utilizam iOS. Ele usa tecnologia de reconhecimento de voz alimentada por IA.
- **Cortana.** A assistente de produtividade pessoal da Microsoft, Cortana usa o mecanismo de busca *Bing* para realizar várias tarefas, incluindo definir lembretes e responder às dúvidas dos usuários.
- **Google Assistente.** Assistente de voz virtual desenvolvido pela Google para dispositivos Android. O Google Assistente pode realizar uma variedade de tarefas, incluindo responder perguntas, ajustar configurações de hardware no dispositivo do usuário, agendar eventos e alarmes.
- **Amazon Alexa.** Utilizado principalmente pela linha de produtos de alto-falantes da Amazon, conhecida como Amazon Echo, Alexa é um serviço de voz baseado em nuvem disponível em mais de 100 milhões de dispositivos Amazon. Em 2019, a Alexa da Amazon era compatível com cerca de 60.000 dispositivos domésticos inteligentes diferentes em todo o mundo, sendo um excelente exemplo de quão popular esses aplicativos se tornaram (LARICCHIA, 2022).
- **Bixby.** O assistente virtual de IA da Samsung funciona principalmente em dispositivos móveis, mas também em algumas geladeiras inteligentes. Bixby pode ser usado para tarefas incluindo mensagens de texto, recuperação de informações meteorológicas, configuração de lembretes de reuniões e leitura de artigos de notícias.
- **Mycroft AI platform.** Mycroft é o primeiro assistente de voz de código aberto que pode ser executado em qualquer plataforma, incluindo desktops, automóveis e até em microcontroladores Raspberry Pi.

Nem todos os assistentes virtuais têm as mesmas habilidades. Eles extraem dados de várias fontes e os coloca em contexto. O processamento avançado de linguagem natural, chamado de Entendimento de linguagem natural (NLU), permite processar o que está sendo dito ou digitado e, em seguida, gera respostas precisas.

Os assistentes virtuais mais avançados são capazes de processar várias tarefas e perguntas complexas para conversar com o usuário de uma maneira amigável e de fácil entendimento. Esses assistentes utilizam IA e *machine learning* para entender e aprender as preferências, podendo até usar isso para fazer previsões e recomendações, com base em ações passadas. Dessa forma, trabalhar com um assistente virtual se torna uma experiência personalizada (ORACLE, 2018).

3 Metodologia da Pesquisa

O objetivo do sistema desenvolvido é controlar ar-condicionados, com a condição que haja total autonomia na forma de controle e programação do dispositivo, garantindo uma maior comodidade ao usuário. Deverá ter também uma integração entre o sistema e Assistentes de voz, aumentando ainda mais a as opções disponíveis para automação. Esse sistema utiliza também sensores de temperatura e umidade, para que exista programações e automações com base nos valores medidos. O controle é dado por um emissor infravermelho que atua diretamente no ar-condicionado. Toda a comunicação entre o usuário e o ar-condicionado é feita via softwares conectados diretamente ao microcontrolador, ou pelo alto-falante inteligente.

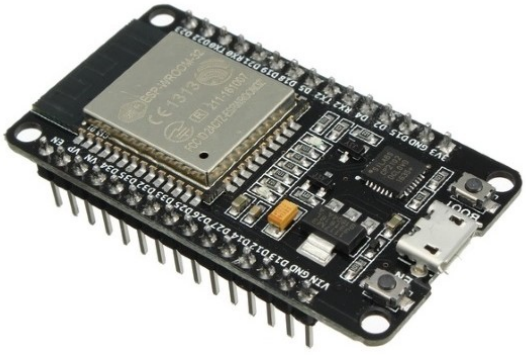
Neste capítulo encontra-se a descrição do projeto desenvolvido, discriminando os recursos físicos e virtuais utilizados, juntamente com a integração entre eles. Na seção "Materiais", é abordado a lista de materiais utilizados para a formação deste trabalho, enquanto que na seção "Metodologia", é explicado como os materiais foram utilizados para cumprir os objetivos propostos, como se deu a integração das partes e como a rotina de testes foi executada.

3.1 Materiais

3.1.1 ESP-32

A ESP32 é uma placa de desenvolvimento que integra o ESP32 SoC (System on Chip) desenvolvida pela empresa *Espressif Systems*, com módulos Wi-Fi e Bluetooth integrado, utilizada em várias aplicações de Internet das Coisas por ser uma opção de baixo custo, mas que oferece versatilidade e confiança para as mais diversas aplicações.

Figura 7 – Representação da Placa de Desenvolvimento ESP32.



Fonte: (MAKERHERO, 2017)

ESP32 foi projetado para aplicativos móveis, eletrônicos vestíveis (Wearables) e Internet das Coisas. Ele apresenta todas as características de última geração de chips de baixo consumo de energia, incluindo vários modos de energia e escalonamento de energia dinâmico. Além disso, a placa disponibiliza várias portas programáveis de entrada e saída (GPIO), memória RAM estática de 520k bytes, RTC (Real Time Clock) com 16Kb de SRAM, aceleradores de hardware (criptografia) como AES, Hash, RSA e ECC, entre outros módulos e periféricos (ESPRESSIF, 2012).

As principais características do ESP32 são:

- Microprocessador Xtensa® single/dual-core LX6 de 32 bits
- Processador pode operar em 80MHz / 160MHz / 240MHz
- Tensão operacional de 3,3
- Alimentação fornecida através do conector Micro USB-B integrado ou diretamente com 5 à 12 VDC através do pino “VIN”
- 25 pinos de Entrada/Saída digitais (DIO)
- 6 pinos de entrada analógica (ADC)
- 2 pinos de saída analógica (DAC)
- 4MB de memória flash
- WiFi nativo padrão 802.11b/g/n
- Bluetooth BLE 4.2 BR/EDR e BLE (Bluetooth Low Energy)

O ESP32 é um dos poucos se não o único controlador de pequeno porte que reúne todas essas propriedades, as quais geram além de uma maneira econômica de se ter uma plataforma embarcada de alta empregabilidade, um enorme avanço para a automatização.

3.1.2 LED emissor de infravermelho

O LED emissor age como atuador no sistema, enviando sinais de comando para o receptor infravermelho no condicionador de ar. Este dispositivo é capaz de converter sinal elétrico em sinal luminoso, realizando as funções escolhidas pelo usuário. Foi usado LED Emissor Infravermelho IR 5mm como mostrado na Figura 8.

Figura 8 – LED Infravermelho.

Fonte: (MAKERHERO, 2023)

3.1.3 Módulo receptor infravermelho 1838T

O sensor 1838T, apresentado na Figura 9, é um componente eletrônico utilizado para receber e decodificar sinais infravermelhos emitidos por controles remotos Pode ser encontrado em TVs, rádios, aparelhos de multimídia e até em aparelhos de ar condicionado. O módulo contém um receptor IR 1838 que conta com tensão de operação entre 2,7 e 5,5V. A comunicação com o microcontrolador se dá por meio de um único pino de saída.

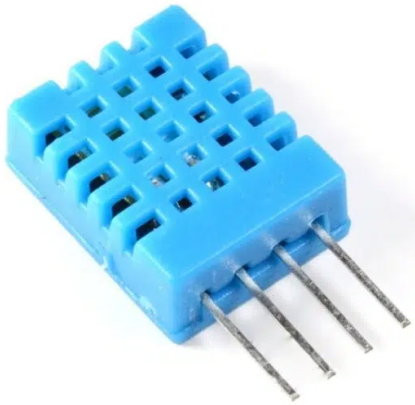
Figura 9 – Receptor Infravermelho 1838T.

Fonte: (RYTRONICS, 2023)

3.1.4 Modulo Sensor de Temperatura e Umidade DHT11

O sensor DHT11 é um dispositivo muito popular utilizado para medir a temperatura e umidade relativa do ar. O sensor DHT11 consiste em um elemento sensor de umidade capacitivo e um termistor de resistência negativa de temperatura (NTC). Esses componentes são combinados em um único pacote, que contém também um circuito integrado que realiza as leituras e fornece os dados de saída digital. O sensor DHT11 possui três pinos de conexão: VCC, DATA e GND.

Figura 10 – Sensor de Umidade e Temperatura DHT11.



Fonte: MAKERHERO, 2023.

Apesar de sua popularidade e baixo custo, o sensor DHT11 possui algumas limitações. A principal delas é a sua precisão nas leituras da temperatura com uma precisão de ±2°C e a leitura da umidade com uma precisão de ±5%. Além disso, a taxa de atualização do sensor é relativamente baixa, o que significa que ele não é adequado para aplicações que exigem uma resposta em tempo real.

3.1.5 Echo Dot 3ª Geração

O Echo Dot é um alto-falante inteligente da Amazon que oferece música, notícias e previsão do tempo por meio de interação por voz com Alexa, assistente virtual da Amzaon, além de compatibilidade com mais de 140 mil dispositivos conectados. Ele possui alguns botões na parte superior que controlam o volume, silenciam o alto-falante e ativam o dispositivo para ouvir uma pergunta ou comando.

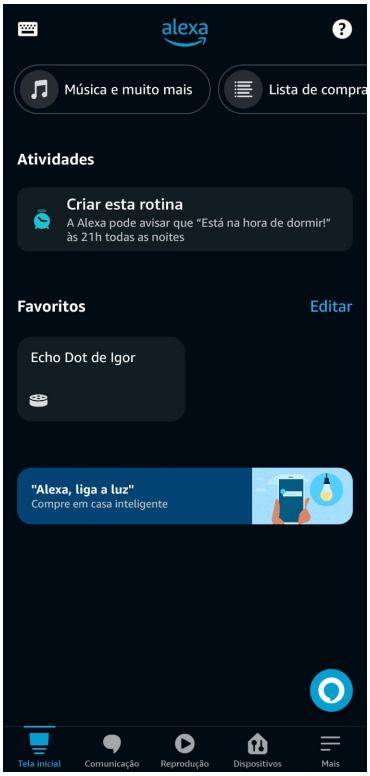
Com Alexa integrada, o aparelho é capaz de realizar uma ampla gama de atividades. Alexa pode tocar música, definir alarmes, controlar outros dispositivos "inteligentes" conectados em casa e responder perguntas pesquisando informações na internet.

Figura 11 – Echo Dot de 3° geração

Fonte: (AMAZON, 2023)

Para conectar aparelhos inteligentes ao alto-falante inteligente, é necessário o aplicativo para smartphones "Alexa", que funciona como interface gráfica para o dispositivo, e é onde são realizadas todas as configurações.

Figura 12 – Interface do aplicativo Alexa



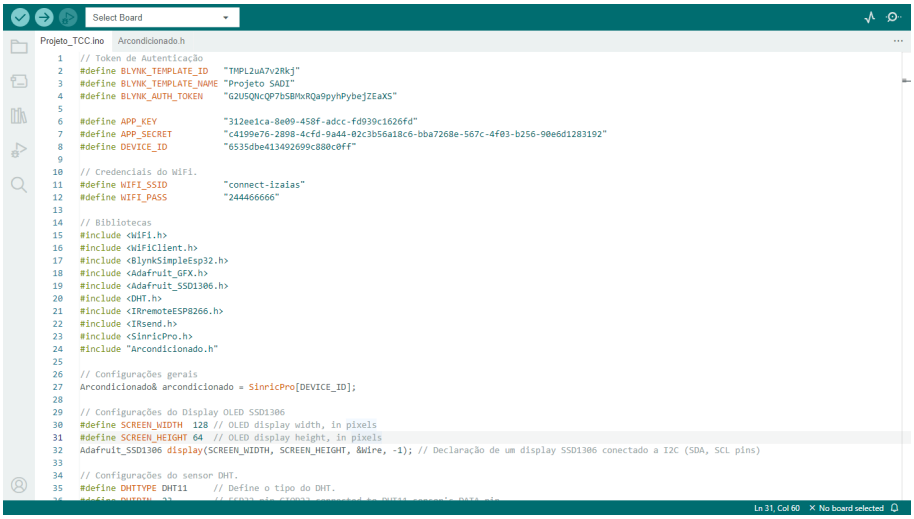
Fonte: Autoria Própria

3.1.6 Arduino IDE

E um ambiente de programação de código aberto, desenvolvido pela organização Arduino, possibilitando o desenvolvimento e carregamento de códigos em placas Arduino e outras compatíveis. A placa ESP32 pode ser programada em várias frameworks diferentes, mais comum sendo a ESP-IDF, framework oficial da *Espressif*, e Arduino IDE, sendo este último em que o microcontrolador foi programado. A interface do Arduino IDE é mostrada na Figura 13.

Além disso, a IDE do Arduino oferece uma vasta biblioteca para diversos tipos de dispositivos, sensores e serviços, incluindo plataformas IoT populares como MQTT, HTTP e TCP, facilitando e simplificando a integração de diferentes sistemas e dispositivos, e assim acelerando o processo de criação e prototipagem de aplicativos IoT.

Figura 13 – Interface do Arduino IDE.

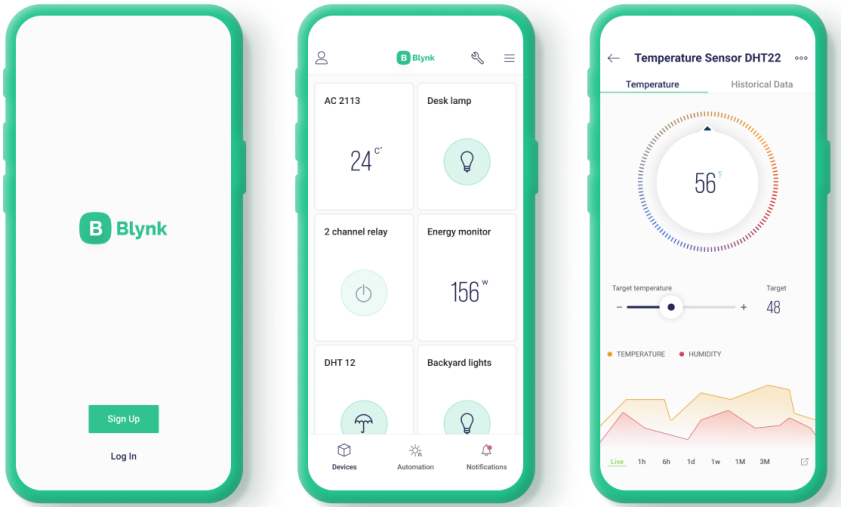


Fonte: Autoria Própria

3.1.7 Blynk IoT

Para o desenvolvimento de uma interface gráfica, utilizamos o software *Blynk*, que é uma plataforma IoT que permite a prototipagem, implantação e gerenciamento remoto de dispositivos eletrônicos conectados em qualquer escala. Se tratando de projetos IoT, o Blynk permite que os usuários conectem seu hardware à nuvem e criem aplicativos para iOS, Android e aplicativos da Web, para que analisem dados históricos e em tempo real de dispositivos, podendo também controlar remotamente, e até receber notificações importantes.

Figura 14 – Interface do Blynk para smartphones.



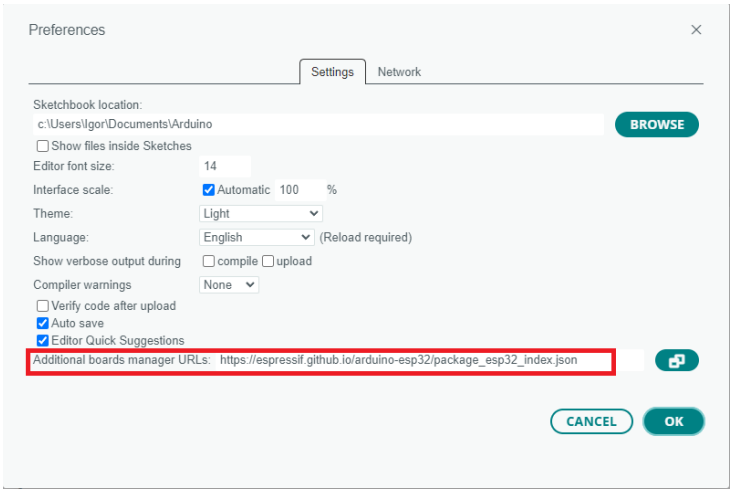
Fonte: Adaptado de (BLYNK, 2023).

3.2 Metodologia

3.2.1 Testes de unidade

Inicialmente para implementação na placa de desenvolvimento ESP32, foi realizada a instalação do Arduino IDE, framework da Arduino na qual tem suporte para o desenvolvimento de aplicações nas placas ESP32. Para utilizar dispositivos ESP32 no arduino é necessário primeiro adicionar as placas ao IDE. Para isso clicamos em Arquivos -> Preferências, e na aba Preferências aberta, escrevemos em "URLs adicionais para Gerenciadores de Placas" a URL *"https://espressif.github.io/arduino-esp32/package_esp32_index.json"*.

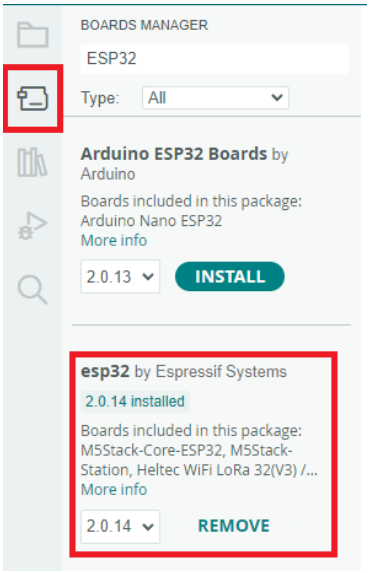
Figura 15 – Configuração para instalar placas ESP32 na IDE do Arduino.



Fonte: Autoria Própria

Após isso vamos em Ferramentas > Placa > Gerenciador de Placas procuramos "ESP32", e a opção de instalar placas de desenvolvimento ESP32 estará disponível. A placa selecionada deverá ser a mesma utilizada para não acorrer problemas de compatibilidade nas GPIOs.

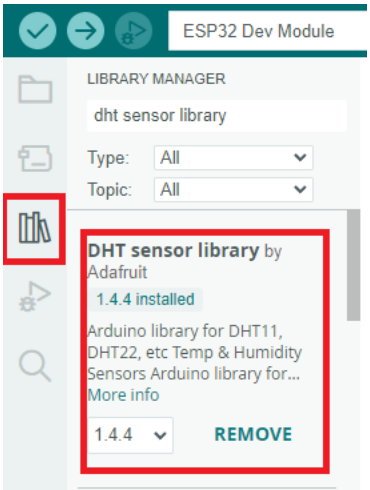
Figura 16 – Configuração para instalar placas ESP32 na IDE do Arduino.



Fonte: Autoria Própria

Com a plataforma instalada, foi iniciado os testes de undiade, onde foram testados todos os componentes individualmente, começando com o ESP32 em conjunto com o sensor de temperatura DHT11. Para isso é necessário a adição das bibliotecas do sensor para que aconteça a comunicação entre os dispositivos. Para adicionar bibliotecas no Arduino IDE, é necessário clicar em "Gerenciador de bibliotecas"e procurar pela biblioteca necessária.

Figura 17 – Configuração para adicionar bibliotecas no Arduino IDE.

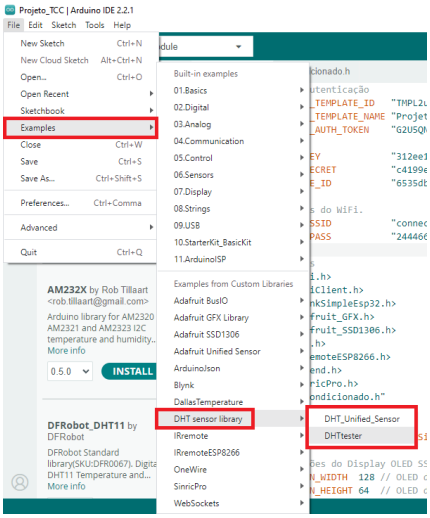


Fonte: Autoria Própria

As bibliotecas instaladas são criadas por usuários, as vezes até por empresas, e são geralmente mantidas atualizadas em relação as mudanças na IDE do Arduino, ou atualizações nas placas de desenvolvimento. As bibliotecas também oferecem códigos

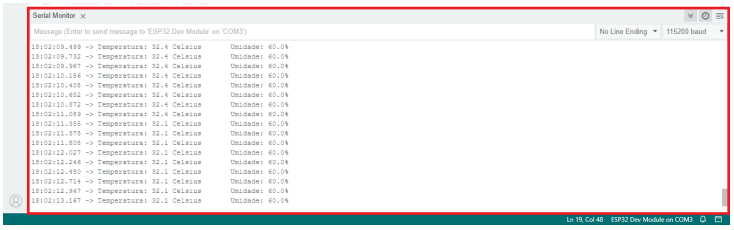
exemplos para as mais diversas funcionalidades. Para o teste do sensor de temperatura foi utilizado o exemplo disponibilizado para testes, como mostrado na figura 18, e os resultados dos testes exibidos no monitor serial são mostrados na figura 19.

Figura 18 – Códigos exemplos disponibilizados pela biblioteca.



Fonte: Autoria Própria

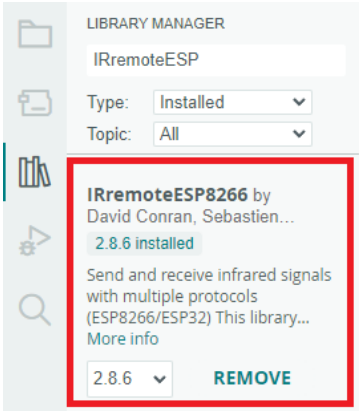
Figura 19 – Resultado do teste de unidade da biblioteca do sensor DHT11 exibidos no monitor serial.



Fonte: Autoria Própria

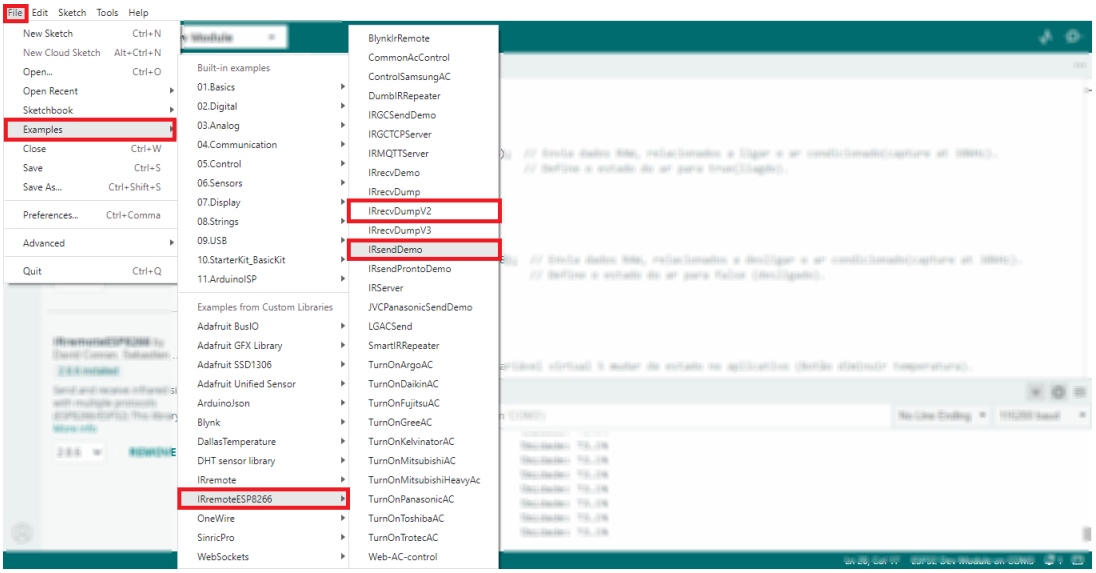
O segundo teste de unidade realizado foi o do módulo receptor infravermelho 1838T e do LED emissor de infravermelho. Para o módulo receptor foi primeiro instalado a biblioteca *IRremoteESP8266*, que fornece suporte para placas de desenvolvimento ESP32. Esta biblioteca permite enviar e receber sinais infravermelhos em um ESP8266 ou ESP32 usando a *framework* do Arduino, com LEDs IR comuns de 940 nm e módulos receptores IR como o 1838T. A biblioteca também conta com exemplos disponíveis com códigos prontos que foram utilizados para os testes.

Figura 20 – Configuração para adicionar a biblioteca *IRremoteESP8266*.



Fonte: Autoria Própria

Figura 21 – Configuração para abrir códigos exemplos da biblioteca.

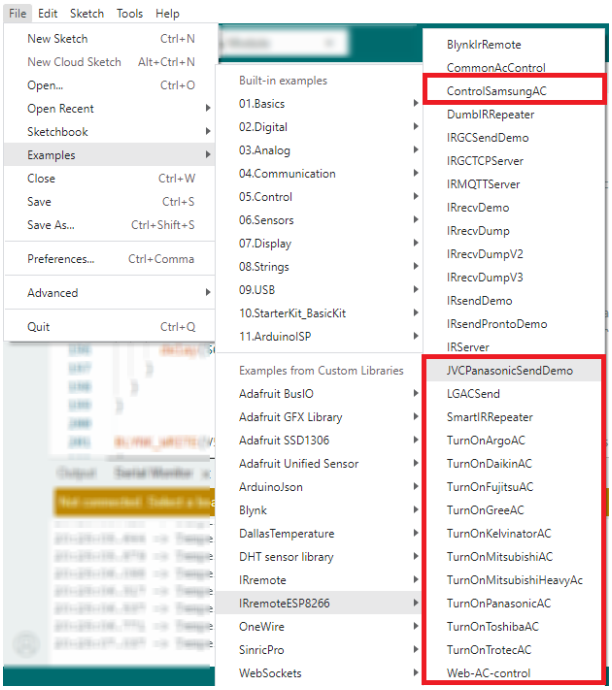


Fonte: Autoria Própria

Com a biblioteca instalada, foi possível fazer a captura dos comandos do controle do ar-condicionado da marca *Elgin*, encontrados no bloco Embedded, localizado na UFCG. Os comandos capturados foram os de ligar e desligar o aparelho, e os comandos relativos as temperaturas de 16°C até 32°C, todos em formato NEC. Com os códigos em formato NEC capturados, foi possível realizar o teste do emissor de infravermelho. A biblioteca *IRremoteESP8266* oferece um exemplo de código para envio de dados IR, como mostra a figura 22. Com ela foi possível fazer a clonagem dos sinais do controle do ar-condicionado, fazendo com que o ar-condicionado identificasse o comando emitido.

Essa biblioteca também conta com um acervo de códigos para controle de de ar-condicionados de diversas marcas como Samsung, Toshiba, Panasonic, Sony e outras, onde os códigos e protocolos para cada fabricante já estão configurados para o uso em cada aparelho que trabalhe nessas configurações. A biblioteca é mantida e atualizada por usuários de forma voluntária, e a cada atualização nova, novos códigos são adicionados, aumentando ainda mais a quantidade de fabricantes disponíveis.

Figura 22 – Códigos prontos para controle de Ar-condicionados de várias fabricantes.



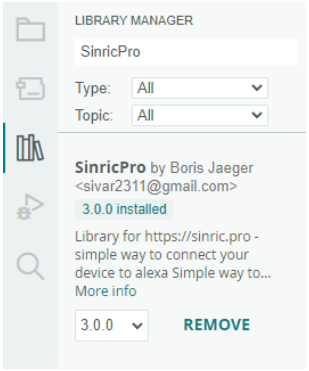
Fonte: Autoria Própria

3.2.2 Conexão entre o ESP32 e Assistente de Voz

Para realizar a conexão entre a placa de desenvolvimento ESP32 e a assistente de voz da Amazon Echo Dot 3, é necessária a adição de uma biblioteca chamada Sinric Pro, que permite conectar e controlar dispositivos IoT como ESP826, ESP3, Raspberry Pi ou Arduino com Amazon Alexa, o Google Home ou NODE-RED de forma gratuita. Com Sinric Pro é possível transformar projetos de microcontrolados em dispositivos reconhecidos por assistentes virtuais e controlá-los por comando de voz.

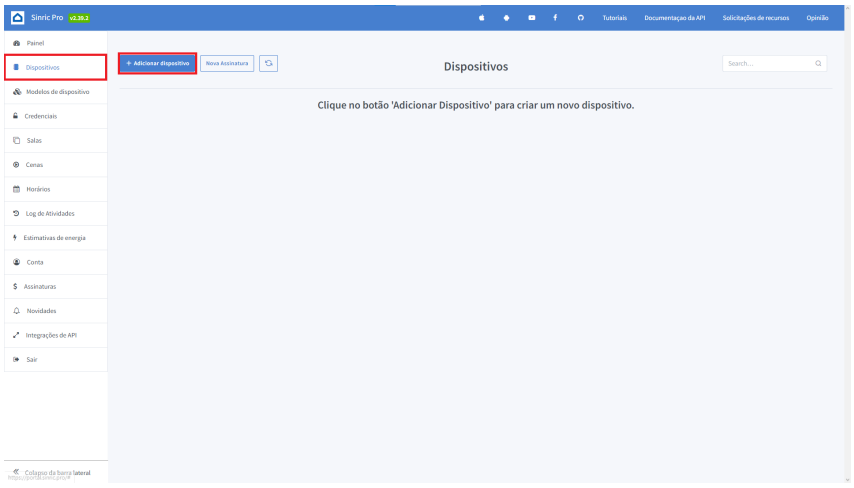
O primeiro passo após instalar a biblioteca, é realizar um cadastro no site da Sinric Pro. Ao entrar no site o usuário poderá adicionar dispositivos, onde podemos escolher o nome do dispositivo, descrição, o tipo do dispositivo que será adicionado, e o local da residência onde se encontra o dispositivo. O usuário também poderá receber notificações quando o dispositivo se conectar, e colocar temporizadores.

Figura 23 – Configuração para adicionar a biblioteca *Sinric Pro*.



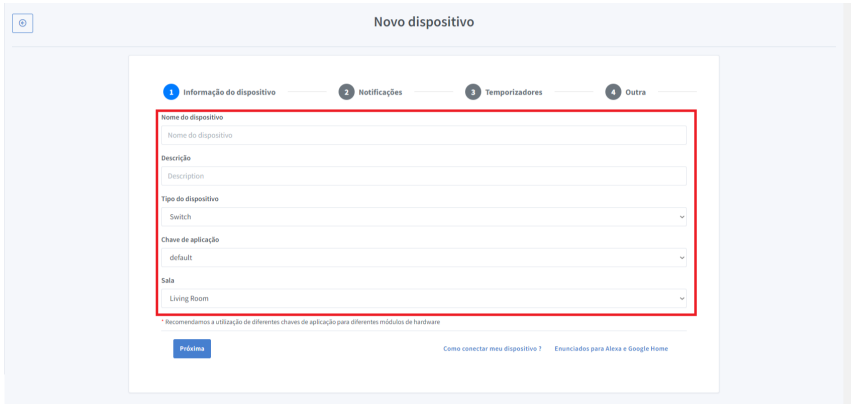
Fonte: Autoria Própria

Figura 24 – Passos para adicionar dispositivo no *Sinric Pro*.



Fonte: Autoria Própria

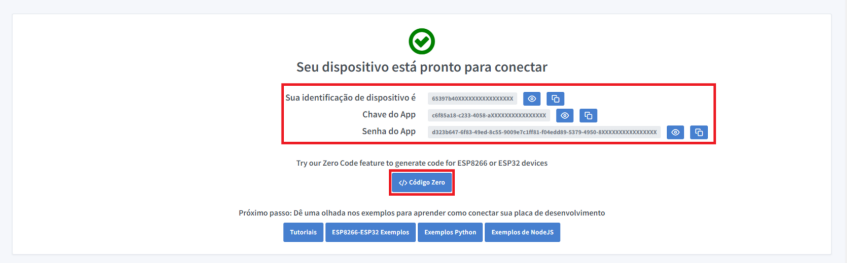
Figura 25 – Passos para adicionar dispositivo no *Sinric Pro*.



Fonte: Autoria Própria

Após a criação do dispositivo, ele está pronto para ser conectado. É disponibilizado algumas informações como chaves de identificação do dispositivo criado, Chave do App, e senha do App, exibidas na Figura 26 utilizadas para fazer a conexão com as funções da biblioteca. Também é disponibilizado a função "Código Zero", onde é possível gerar o código para o dispositivo criado, onde já estão prontas todas as funções necessárias, com as chaves de acesso já disponíveis.

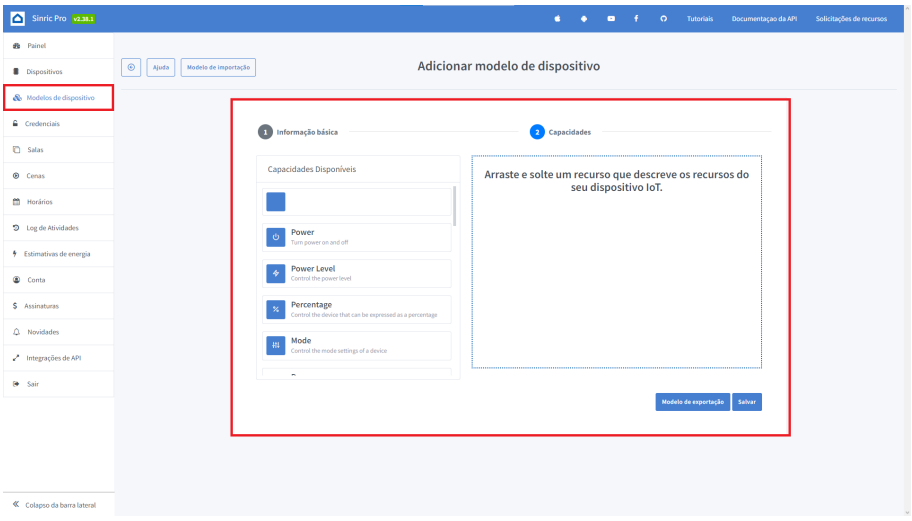
Figura 26 – Janela com informações após a criação do dispositivo no *Sinric Pro*.



Fonte: Autoria Própria

Inicialmente foi criado um dispositivo predefinido pelo próprio Sinric Pro, chamado "Ar condicionado", onde ele tem algumas funções básicas de ar-condicionados, como por exemplo ligar, desligar, escolher temperatura e selecionar modos. Também é possível criar dispositivos personalizados na aba "Modelos de dispositivo", onde é possível escolher individualmente as capacidades que o dispositivo criado terá, como mostrado na figura 27. Para testar a funcionalidade foi criado outro dispositivo chamado "ESP32", onde foram escolhidas as funcionalidades de ligar e desligar, sensor de temperatura, e termostato para controle da temperatura.

Figura 27 – Configuração para criar dispositivos personalizados no *Sinric Pro*.



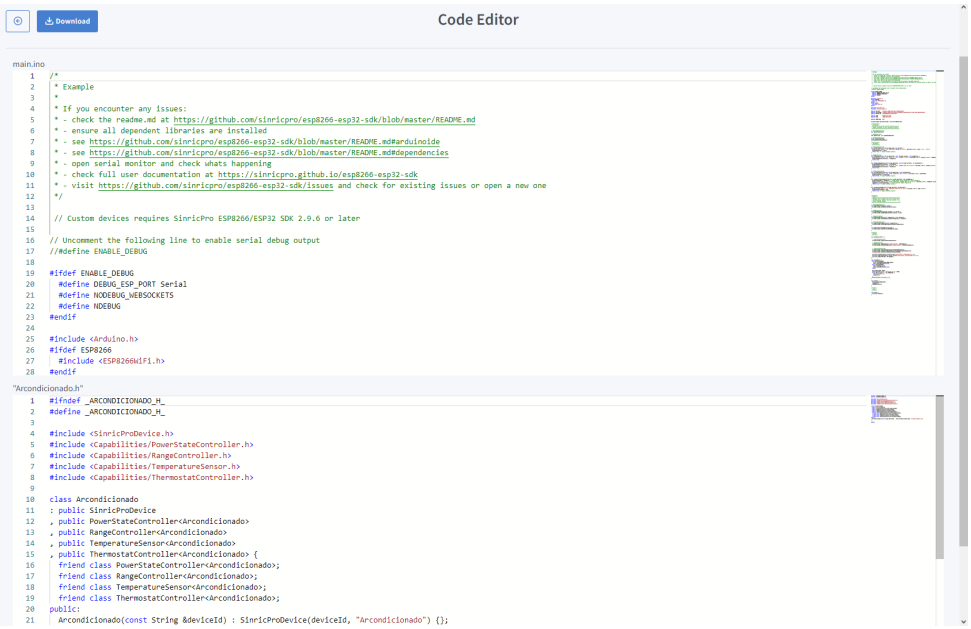
Fonte: Autoria Própria

Com a criação de dispositivos personalizados, temos a opção de gerar o código de forma automática contendo todos as funções necessárias para as capacidades do dispositivo, como mostrado na Figura 29. O código gerado foi então copiado e colado na IDE do Arduino.

Figura 28 – Painel com os dispositivos criados no *Sinric Pro*.

Fonte: Autoria Própria

Figura 29 – Painel com os códigos gerados.



Fonte: Autoria Própria

As funções mostradas na Figura 30 correspondem a uma função disponível no assistente de voz, como por exemplo, pedir por comando de voz para a Alexa ligar ou desligar o dispositivo, mudar os valores de temperatura do termostato, mudar os modos de operação, ou ler os valores de temperatura.

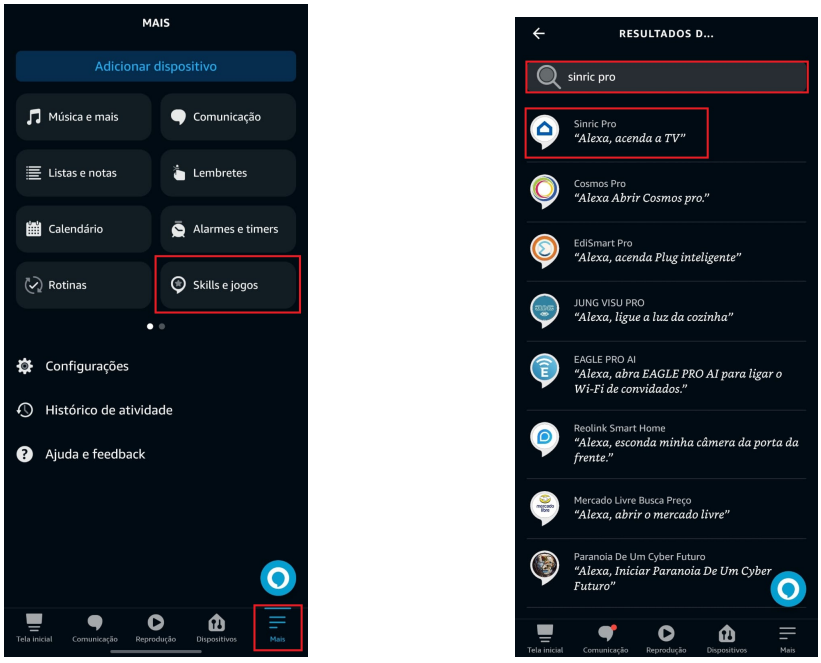
Figura 30 – Funções geradas para a integração com Alexa.

```
/*************
 * Callbacks *
 *************/

// PowerStateController
bool onPowerState(const String &deviceId, bool &state) {
  Serial.printf("[Device: %s]: Powerstate changed to %s\r\n", deviceId.c_str(), state ? "on" : "off");
  globalPowerState = state;
  return true; // request handled properly
}

// RangeController
bool onRangeValue(const String &deviceId, const String& instance, int &rangeValue) {
  Serial.printf("[Device: %s]: Value for \"%s\" changed to %d\r\n", deviceId.c_str(), instance.c_str(), rangeValue);
  globalRangeValues[instance] = rangeValue;
  return true;
}

// ThermostatController
bool onTargetTemperature(const String &deviceId, float &targetTemp) {
  Serial.printf("[Device: %s]: Target temperature set to %f\r\n", deviceId.c_str(), targetTemp);
  globalTargetTemp = targetTemp;
  return true; // request handled properly
}

bool onThermostatMode(const String& deviceId, String& mode) {
  Serial.printf("[Device: %s]: Thermostat mode set to %s\r\n", deviceId.c_str(), mode.c_str());
  globalThermostatMode = mode;
  return true; // request handled properly
}
```

Fonte: Autoria Própria

Feito isso, o dispositivo está pronto para ser identificado pela Alexa. O próximo passo é instalar o aplicativo Sinric Pro no próprio aplicativo da Alexa, já mostrado na Figura 12. Lá é oferecido aplicativos que fazem a integração com os dispositivos das marcas correspondentes.

Figura 31 – Baixar Sinric Pro no aplicativo da Alexa.

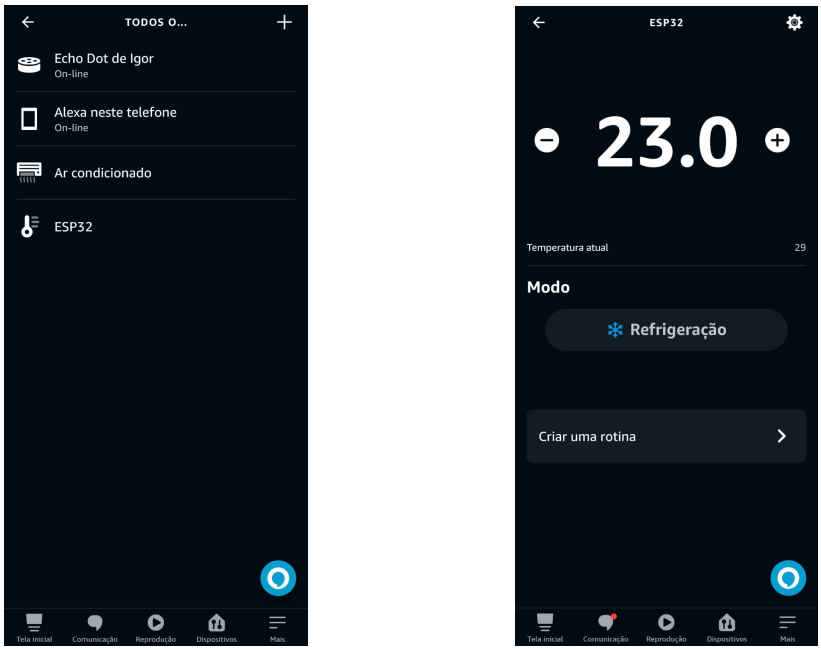


(a) (b)

Fonte: Autoria Própria

Com o aplicativo Sinric Pro instalado, agora o assistente de voz é capaz de identificar os dispositivos cadastrados e controlá-los por meio dos comandos de voz enviados. Os dispositivos reconhecidos pela assistente de voz são mostrados na figura 32.

Figura 32 – Dispositivos 'Ar condicionado' e 'ESP32' reconhecidos no painel da Alexa.



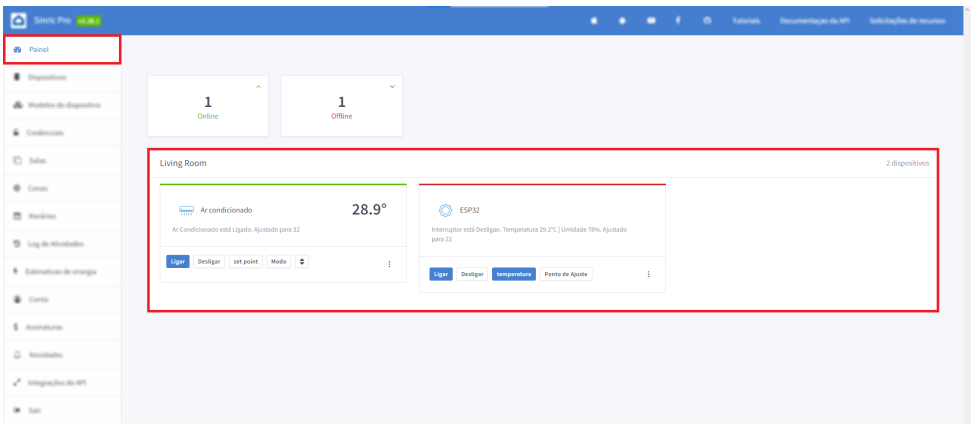
(a) Todos dispositivos encontrados.

(b) Painel de controle do dispositivo "ESP32".

Fonte: Autoria Própria

No painel do site do Sinric Pro também é possível visualizar o status dos dispositivos conectados, assim como as informações obtidas pelo dispositivo, como mostrado na Figura 33.

Figura 33 – Painel com os dispositivos criados no *Sinric Pro*.

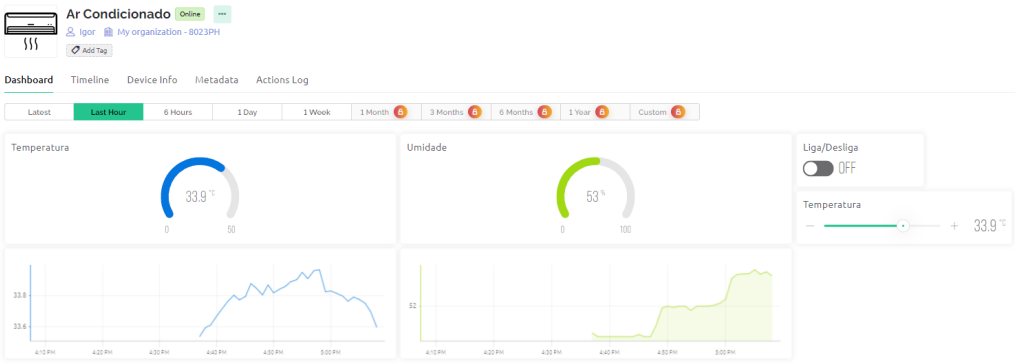


Fonte: Autoria Própria

3.2.3 Conexão entre o ESP32 e *Blynk*

Para poder fazer a integração do aplicativo *Blynk* com o ESP32, seguimos os mesmos passos do Sinric Pro. Primeiro é necessário a criação de uma conta em seu site para ser possível criar um novo projeto e conectá-lo ao microcontrolador. É disponibilizado alguns exemplos de projetos, chamados *Blueprints*, que pode facilitar na criação do projeto. Com o projeto criado, podemos definir as variáveis de temperatura e umidade do sensor DHT11 que serão enviadas para o aplicativo do Blynk, e montar um *dashboard* personalizado para exibição dos resultados de temperatura de umidade em tempo real, como mostrado na Figura 34.

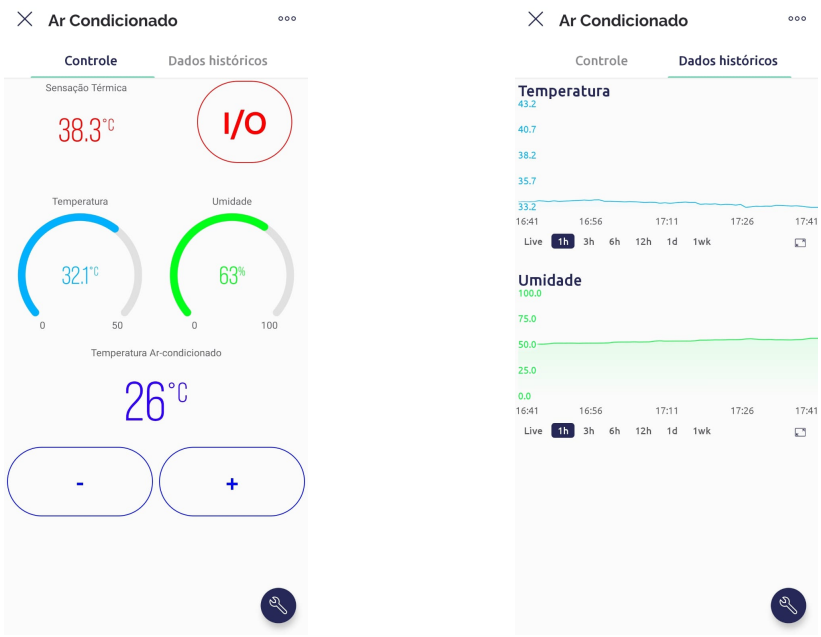
Figura 34 – Dashboard criado para o projeto no Blynk.



Fonte: Autoria Própria

Apesar disso, o foco princial do Blynk é o seu aplicativo para smartphones, onde existe uma variedade maior de itens a serem escolhidos para a interface gráfica, e a opção de utilizar botões para servir como atuadores no microcontrolador. Para isso, foi instalado o aplicativo Blynk, que pode ser encontrado em qualquer loja de aplicativos para smartphones, foi conectada com a conta já criada anteriormente, e como o projeto já foi criado no site, é necessário apenas a criação do dashboard para celulares, como mostrado na Figura 35.

Figura 35 – Dashboards do Blynk para smartphones.



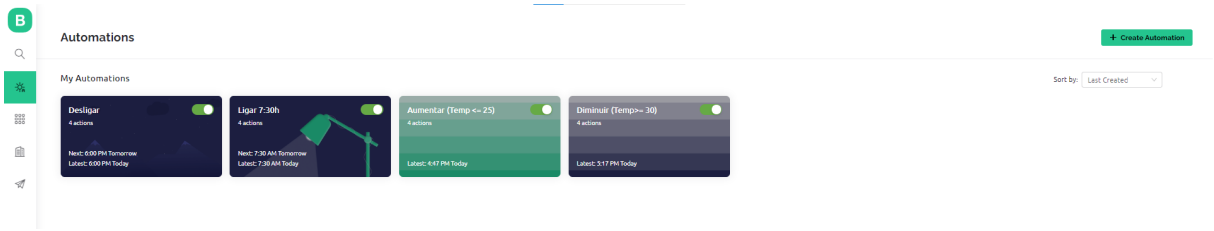
(a) Painel principal

(b) Painel com histórico de dados

Fonte: Autoria Própria

Além disso, o Blynk também permite a criação de automações, que fazem o sistema realizar uma ação de acordo com alguma condição indicada. Foi criada condições para ligar automaticamente o ar-condicionado quando for 07:30h da manhã e desligar às 18:00h, e também de aumentar a temperatura do ar-condicionado quando a temperatura do ambiente estiver abaixo de 25°C e diminuir quando estiver maior que 30°C, como mostra a Figura 36.

Figura 36 – Painel para criação de automações no *Blynk*.



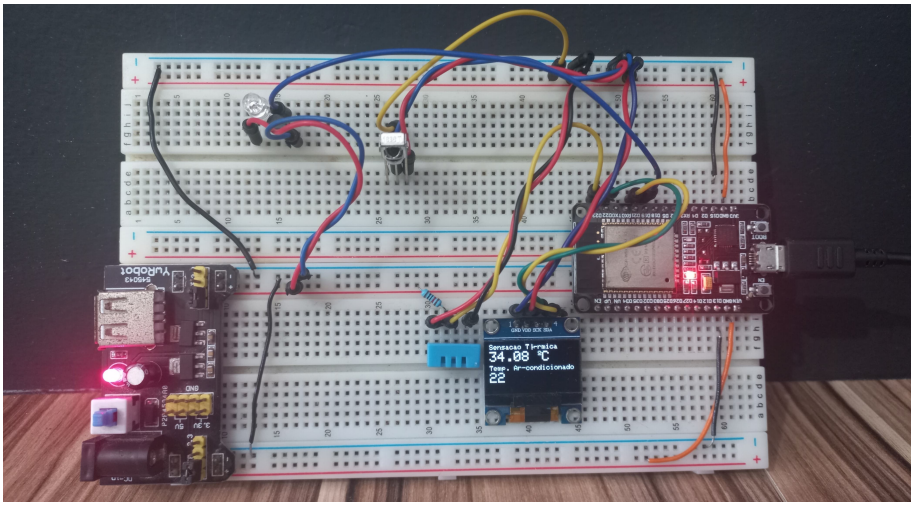
Fonte: Autoria Própria

3.2.4 Testes de integração

Após realizar os testes de unidade em cada material e biblioteca separadamente, foi realizado o teste de integração. Para isso todos os módulos e bibliotecas foram combinados e testados em grupo, onde foram realizados testes para verificar conflitos entre eles. A Figura 37 mostra o protótipo feito para os testes.

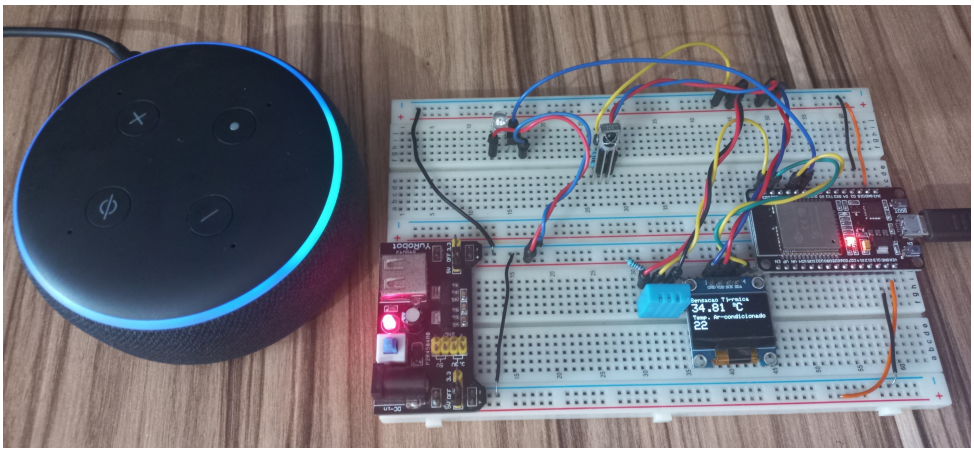
Foram adicionados alguns componentes como um display OLED SSD1306 para exibir os resultados de temperatura e qual valor está selecionado no dispositivo de ar-condicionado. Também foi adicionado um transistor BC337 do tipo NPN para ampliar a distância de sinais enviados pelo LED infravermelho e um módulo de fonte ajustável com saída de 3.3v ou 5v, ampliando as opções de alimentação do ESP32. O arranjo do protótipo com o alto-falante inteligente echo dot é mostrado na Figura 38.

Figura 37 – Protótipo do projeto realizado.



Fonte: Autoria Própria

Figura 38 – Protótipo do projeto realizado com o assistente de voz.



Fonte: Autoria Própria

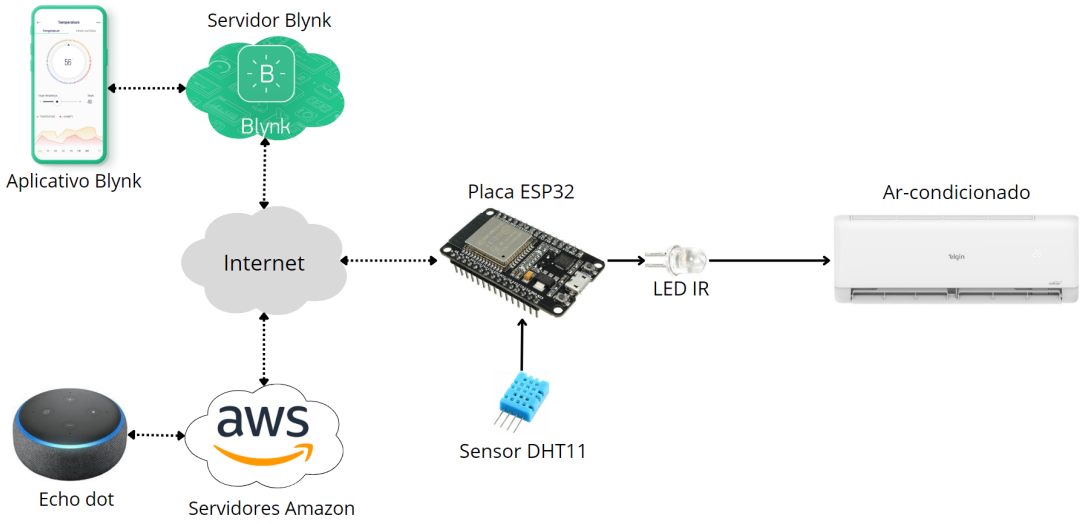
Com todos os módulos integrados, o código, encontrado no Anexo A, foi compilado no ESP32 por meio do cabo USB conectado ao computador. Com isso temos nosso protótipo, que pode enviar sinais infravermelho por meio do aplicativo de celular, onde temos uma interface gráfica para analisar os dados de temperatura e umidade e controlar o dispostivo, ou por meio de voz com o uso do echo dot conectado ao dispositivo por meio da internet.

4 Resultados e discussões

Aplicando os métodos e materiais presentes neste trabalho, foi possível realizar a implementação de um sistema que monitora e controla pela internet equipamentos de ar-condicionados, por meio de comunicação infravermelha, com o auxilio de assistentes de voz utilizando uma placa de desenvolvimento ESP32.

Os testes iniciais realizados de forma individual, bem como os testes com os módulos integrados na placa ESP32 ocorreram de forma satisfatória. No entanto, foi necessária a utilização de um transistor para ampliar o alcance do sinal enviado pelo LED infravermelho, garantindo que houvesse uma melhor resposta no controle do ar-condicionado a distancia. Uma ilustração do projeto desenvolvido é mostrado na Figura 39.

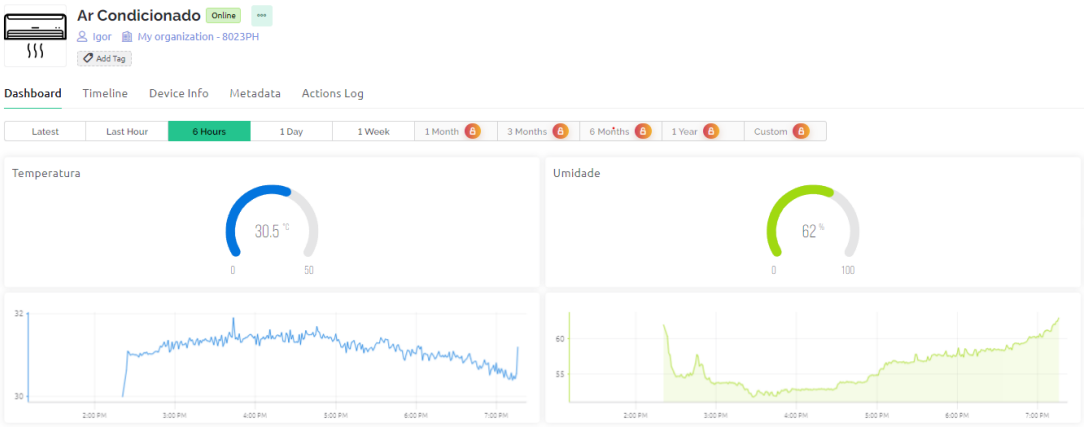
Figura 39 – Ilustração do projeto realizado.



Fonte: Autoria Própria

Utilizando o sensor DHT11 conectado ao microcontrolador, foi possível realizar o monitoramento da temperatura real no ambiente do ar-condicionado, para que houvesse comparações entre as temperaturas reais e a que o equipamento exibe. Os dados de temperatura e umidade capturados pelo dispositivo podem ser visualizados pelo dashboard criado no Blynk, onde um *timestamp* dos valores é exibido, como mostra a Figura 40.

Figura 40 – Painel para monitoramento da temperatura e umidade capturada pelo DHT11.



Fonte: Autoria Própria

Para o controle foi feito no aplicativo do Blynk mostrado na seção 3.2.3 e figura 35, que conta com botões para atuações de ligar e desligar o ar-condicionado, como também aumentar ou diminuir a temperatura. Também para o controle temos o Echo dot, que com o Sinric Pro pro foi possível fazer a integração com o ESP32 e enviar os sinais infravermelhos de comando para o ar-condicionado. No site do Sinric Pro é possível visualizar um log de atividades, exibindo os comandos de voz feito para Alexa que ocasionaram em alguma ação no microcontrolador. O log de atividades pode ser visto na figura 41.

Figura 41 – Log de atividades exibindo as ações do Echo dot sobre o ESP32.

Log de Atividades

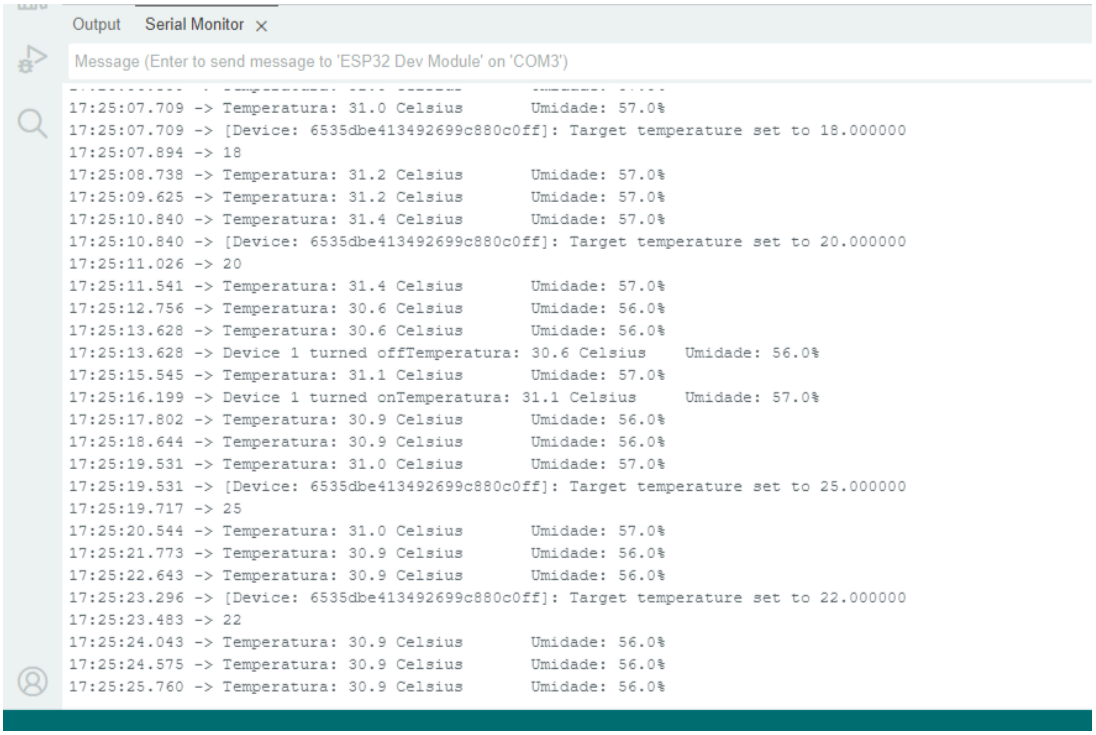
Últimos 200 registros dos últimos 3 dias

IP ADDRESS	LOCALIZAÇÃO	NOME DO APP	ATIVIDADE	TIMESTAMP
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 23	2023-10-29 17:23:59
alexa.amazon.com	-	alexa-skill	[Ar condicionado] turned on	2023-10-29 17:23:47
alexa.amazon.com	-	alexa-skill	[Ar condicionado] turned off	2023-10-29 17:23:35
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 24	2023-10-29 17:22:50
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 16	2023-10-29 17:22:37
alexa.amazon.com	-	alexa-skill	[Ar condicionado] turned on	2023-10-29 17:21:46
alexa.amazon.com	-	alexa-skill	[Ar condicionado] turned off	2023-10-29 17:21:24
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 25	2023-10-29 17:21:08
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 32	2023-10-29 17:20:38
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 16	2023-10-29 17:20:20
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 24	2023-10-29 17:20:09
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 25	2023-10-29 17:19:54
170.80.232.239	Queimadas, Brazil (BR)	websocket	[Ar condicionado] connected	2023-10-29 17:14:56
170.80.232.239	Queimadas, Brazil (BR)	websocket	Login via App key: [default]	2023-10-29 17:14:53
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 22	2023-10-29 17:12:45
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 22	2023-10-29 17:07:17
alexa.amazon.com	-	alexa-skill	[Ar condicionado] set the temperature to 25	2023-10-29 17:06:52
170.80.232.239	Queimadas, Brazil (BR)	websocket	[Ar condicionado] connected	2023-10-29 17:06:31

Fonte: Autoria Própria

Também é possível acompanhar as ações realizadas no Monitor Serial do Arduino IDE, que exibem os valores de temperatura e umidade atuais, assim como as ações realizadas pelo Echo dot no ESP32, como mostra a figura 42.

Figura 42 – Monitor serial exibindo todas as ações feitas no ESP32.



Fonte: Autoria Própria

Contudo, o dispositivo projetado tem capacidade para controlar apenas o ar-condicionado da marca *Elgin* usado para testes, já que apenas foram capturados os códigos de sinais do controle do mesmo. Para ar-condicionados de diferentes marcas e modelos, é necessária a captura dos sinais utilizando a biblioteca *IRremoteESP8266*, assim como mostrado na subseção 3.2.1.

Dessa forma, uma sugestão de melhoria seria a criação de um banco de dados contendo os sinais capturados de várias fabricantes e modelos diferentes encontrados pela UFCG, que possuem diferentes protocolos de comunicação, e que pudesse haver uma escolha entre os fabricantes, tornando mais fácil o controle de vários equipamentos, precisando apenas da seleção correta entre marca e modelo necessário.

5 Considerações finais

Esse trabalho teve como objetivo o desenvolvimento de um sistema para monitoramento e controle de ar-condicionados, utilizando métodos já conhecidos, bem como o uso de assistentes de voz como método alternativo de controle. Diante do que foi proposto neste trabalho, foi possível realizar o desenvolvimento de um dispositivo de fácil uso utilizando microcontrolador ESP32 para automatizar o controle pela internet com a finalidade de garantir, com eficiência energética, maior comodidade aos usuários, conforme foi definido como objetivo.

Com esse projeto é possível fazer o controle de qualquer tipo de condicionadores de ar ou quaisquer outros tipos de aparelhos que operem mediante sinais enviados em infravermelho. Para isso, basta utilizar um circuito com módulo receptor de infravermelho adequado para captar e decodificar os respectivos códigos e protocolos.

Como trabalhos futuros, planeja-se uma continuidade no projeto e ampliar a biblioteca de protocolos de diferentes fabricantes para garantir o uso do projeto em diversos tipos de ar-condicionados. Além disso, a criação de um circuito separado para a captação de sinais infravermelho, para facilitar na ampliação da biblioteca de códigos e disponibiliza-la online, a fim de contribuir em eventuais projetos futuros. Por fim, é possível agregar o trabalho com a implementação de ideias de projetos similares e produtos reais com finalidades similares, além de expandir o uso para outros tipos de equipamentos, como janelas, cortinas e portas inteligentes, e testá-los em aplicações reais.

Referências Bibliográficas

ALTIUM. Infrared communication concepts. 2017. Acesso em: 04 de out. de 2023. Disponível em: <https://techdocs.altium.com/display/FPGA/Infrared+Communication+Concepts>. Citado 3 vezes nas páginas 6, 7 e 8.

ALTIUM. Nec infrared transmission protocol. 2017. Acesso em: 04 de out. de 2023. Disponível em: <https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>. Citado 2 vezes nas páginas 6 e 8.

AMAZON. Echo dot (3ª geração). Acesso em: 05 de out. de 2023. 2023. Disponível em: <https://www.amazon.com.br/Echo-Dot-3%C2%AA-Gera%C3%A7%C3%A3o-Cor-Preta/dp/B07PDHSJ1H/ref=cm_cr_arp_d_product_top?ie=UTF8>. Citado na página 13.

ARCHTOOLBOX. How air conditioners work. 2021. Acesso em: 03 de out. de 2023. Disponível em: <https://www.archtoolbox.com/how-air-conditioners-work/>. Citado na página 3.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128610001568>. Citado na página 1.

BLYNK. Blynk documentation. Acesso em: 15 de out. de 2023. 2023. Disponível em: <https://docs.blynk.io/en/>. Citado na página 15.

DAIKIN. The anatomy of an air conditioner. *Buyers Guide*, 2022. Acesso em: 03 de out. de 2023. Disponível em: <https://www.daikin.com.au/articles/buyers-guide/anatomy-air-conditioner>. Citado 2 vezes nas páginas 3 e 4.

ENGINEERING, S. S. A. C. Commercial air ventilation system (hvac). 2023. Acesso em: 03 de out. de 2023. Disponível em: <https://www.safetyaircond.com.my/commercial-air-ventilation-hvac-malaysia/>. Citado na página 4.

EPE. Empresa de pesquisa energética. *Uso de Ar Condicionado no Setor Residencial Brasileiro: Perspectivas e contribuições para o avanço em eficiência energética*, 2018. Acesso em: 04 de ago. de 2023. Disponível em: <https://www.epe.gov.br/sites-pt/publicacoes-dadosabertos/publicacoes/PublicacoesArquivos/publicacao-341/NT%20EPE%20030_2018_18Dez2018.pdf>. Citado na página 1.

EPE. Empresa de pesquisa energética. *Relatório Síntese do Balanço Energético Nacional*, 2023. Acesso em: 04 de ago. de 2023. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-2023>. Citado na página 1.

ESPRESSIF. Esp32 series. *Espressif Systems*, 2012. Acesso em: 05 de ago. de 2023. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Citado na página 11.

HALL, M. R. *Materials for energy efficiency and thermal comfort in buildings*. Cambridge, England: Woodhead Publishing, 2010. (Woodhead Publishing Series in Energy). Citado na página 3.

HUNDY, G. F.; TROTT, A. R.; WELCH, T. C. Fundamentals. In: *Refrigeration, Air Conditioning and Heat Pumps*. [S.l.]: Elsevier, 2016. p. 1–18. Citado na página 4.

ISLAM, M. T. et al. Iot enabled virtual home assistant using raspberry pi. In: _____. [S.l.: s.n.], 2022. p. 559–570. ISBN 978-981-19-2280-0. Citado na página 8.

KHANCHUEA, K.; SIRIPOKARPIROM, R. A multi-protocol iot gateway and wifi/ble sensor nodes for smart home and building automation: Design and implementation. In: *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. [S.l.: s.n.], 2019. p. 1–6. Citado na página 1.

KITABAYASHI, H. et al. Development of high power infrared led. *SEI Technical Review*, p. 71–74, Abril 2010. Citado na página 5.

LARICCHIA, F. Number of digital voice assistants in use worldwide from 2019 to 2024 (in billions). *Statista*, Março 2022. Acesso em: 05 de out. de 2023. Disponível em: <https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/>. Citado 2 vezes nas páginas 8 e 9.

LV, J. et al. A new usb home appliances based on pc and infrared remote control protocol. In: *2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*. [S.l.: s.n.], 2010. v. 3, p. 572–575. Citado 3 vezes nas páginas 5, 6 e 7.

MAIER, A.; SHARP, A.; VAGAPOV, Y. Comparative analysis and practical implementation of the esp32 microcontroller module for the internet of things. In: *2017 Internet Technologies and Applications (ITA)*. [S.l.: s.n.], 2017. p. 143–148. Citado na página 1.

MAKERHERO. Módulo wifi esp32 bluetooth. Acesso em: 05 de out. de 2023. 2017. Disponível em: <https://www.makerhero.com/produto/modulo-wifi-esp32-bluetooth/>. Citado na página 10.

MAKERHERO. Led emissor infravermelho ir 5mm. Acesso em: 05 de out. de 2023. 2023. Disponível em: <https://www.makerhero.com/produto/led-emissor-infravermelho-ir-5mm/>. Citado na página 12.

ORACLE. What is a digital assistant? 2018. Acesso em: 05 de out. de 2023. Disponível em: <https://www.oracle.com/chatbots/what-is-a-digital-assistant/>. Citado na página 9.

RYTRONICS. Tsop1838 ir receiver. Acesso em: 05 de out. de 2023. 2023. Disponível em: <https://www.rytronics.in/product/tsop1838-ir-receiver/>. Citado na página 12.

SEMICONDUCTORS, V. Data formats for ir remote control. 2019. Acesso em: 04 de out. de 2023. Disponível em: <https://www.vishay.com/docs/80071/dataform.pdf>. Citado 2 vezes nas páginas 6 e 7.

YASAR, K.; BOTELHO, B. virtual assistant (ai assistant). *TechTarget*, 2023. Acesso em: 05 de out. de 2023. Disponível em: <https://www.techtarget.com/searchcustomerexperience/definition/virtual-assistant-AI-assistant>. Citado na página 8.

ANEXO A – Código Implementado

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

```
// Token de Autentica  o
#define BLYNK_TEMPLATE_ID   "Template ID"
#define BLYNK_TEMPLATE_NAME "Nome do Template"
#define BLYNK_AUTH_TOKEN    "Token de Autentica  o"

#define APP_KEY             "KEY"
#define APP_SECRET          "APP SECRET"
#define DEVICE_ID           "DEVICE ID"

// Credenciais do WiFi.
#define WIFI_SSID           "ID WIFI"
#define WIFI_PASS           "Senha WIFI"

// Bibliotecas
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <IRremoteESP8266.h>
#include <IRsend.h>
#include <SinricPro.h>
#include "Arcondicionado.h"
#include "Codigos.h"

// Configura  es gerais
Arcondicionado& arcondicionado = SinricPro[DEVICE_ID];

// Configura  es do Display OLED SSD1306
#define SCREEN_WIDTH  128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1); //
    Declara  o de um display SSD1306 conectado a I2C (SDA, SCL pins)

// Configura  es do sensor DHT.
#define DHTTYPE DHT11     // Define o tipo do DHT.
#define DHTPIN  23        // ESP32 pin GIOP23 connected to DHT11 sensor's
    DATA pin.
DHT dht(DHTPIN, DHTTYPE); // declara  o do DHT conectado no pino DHTPIN,
    do tipo DHTTYPE.
```

```
39 float Temp;                  // vari vel para armazenar Temperatura em
       Celsius.
40 float Humidity;              // vari vel para armazenar a humidade em %.
41 float HITemp;                // vari vel para armazenar a sensa  o t rmica
       em Celcius.
42
43 // Variavies relacionadas ao Ar condicionado
44 bool estado = false;              // Estado do ar condicionado (Ligado/
       Desligado).
45 int TempAC = 25;                     // Temperatura do Ar condicionado.
46 const uint16_t GPIOIrLed = 19;      // ESP32 GPIO utilizado pelo led
       infravermelho.
47
48 IRsend irsend(GPIOIrLed);           // Confiura o pino que vai ser utilziado
       para enviar a mensagem.
49
50 // Task Handle da fun  o dos sensores. Faz a fun  o sensores funcionar
       simultaneamente em outra parte do Core do ESP32.
51 TaskHandle_t Sensor_Task;
52
53 // This function creates the timer object. It's part of Blynk library
54 BlynkTimer timer;
55
56 void myTimer()
57 {
58   // This function describes what will happen with each timer tick
59   // timer    criado para impedir que aconte a uma sobrecarga no servidor
60   Blynk.virtualWrite(V0,Temp);       // vari vel Temp    alocada na
       vari vel virtual V0
61   Blynk.virtualWrite(V1,Humidity);  // vari vel Humidity    alocada na
       vari vel virtual V1
62   Blynk.virtualWrite(V3,TempAC);     // vari vel TempAC    alocada na
       vari vel virtual V4
63   Blynk.virtualWrite(V4,HITemp);     // vari vel HITemp    alocada na
       vari vel virtual V4
64   Serial.printf("Temperatura: %2.1f Celsius\tUmidade: %2.1f%%\r\n", Temp,
       Humidity);
65 }
66
67 BLYNK_WRITE(V2) // Executa quando a vari vel virtual V2 mudar de estado no
       aplicativo (Bot o liagr/desliar).
68 {
69   if(param.asInt() == 1) // Caso V2 mude para 1
70   {
71     if(estado == false)
72     {
73       TempAC = 18;
```

```
74        Serial.println("Ligando");
75        irsend.sendRaw(rawData, 221, 38);  // Envia dados RAW, relacionados a
      ligar o ar condicionado(capture at 38kHz).
76        estado = 1;                        // Define o estado do ar para true
      (liagdo).
77        delay(500);
78      }
79      else
80      {
81        Serial.println("Desligando");
82        irsend.sendRaw(rawData0, 221, 38);  // Envia dados RAW, relacionados
      a desligar o ar condicionado(capture at 38kHz).
83        estado = 0;                         // Define o estado do ar para
      false (desligado).
84        delay(500);
85      }
86    }
87 }
88
89 BLYNK_WRITE(V5) // Executa quando a vari vel virtual 5 mudar de estado no
      aplicativo (Bot o diminuir temperatura).
90 {
91    if(param.asInt() == 1)
92    {
93      TempAC --;
94      controle();
95      Serial.println(TempAC);
96    }
97 }
98
99 BLYNK_WRITE(V6) // Executa quando a vari vel virtual 5 mudar de estado no
      aplicativo (Bot o aumentar temperatura).
100 {
101    if(param.asInt() == 1)
102    {
103      TempAC ++;
104      controle();
105      Serial.println(TempAC);
106    }
107 }
108
109 //Fun  o que contem todos as possiveis temperaturas de envio para os
      dados RAW
110 void controle()
111 {
112    switch (TempAC)
113    {
```

```
114     case 16:  // Caso que TempAC for 16  , envia o dado RAW relacionado a
        16
115       irsend.sendRaw(rawData16, 221, 38);  // Send a raw data capture at 38
        kHz.
116     break;
117
118     case 17:
119       irsend.sendRaw(rawData17, 221, 38);  // Send a raw data capture at 38
        kHz.
120     break;
121
122     case 18:
123       irsend.sendRaw(rawData18, 221, 38);  // Send a raw data capture at 38
        kHz.
124     break;
125
126     case 19:
127       irsend.sendRaw(rawData19, 221, 38);  // Send a raw data capture at 38
        kHz.
128     break;
129
130     case 20:
131       irsend.sendRaw(rawData20, 221, 38);  // Send a raw data capture at 38
        kHz.
132     break;
133
134     case 21:
135       irsend.sendRaw(rawData21, 221, 38);  // Send a raw data capture at 38
        kHz.
136     break;
137
138     case 22:
139       irsend.sendRaw(rawData22, 221, 38);  // Send a raw data capture at 38
        kHz.
140     break;
141
142     case 23:
143       irsend.sendRaw(rawData23, 221, 38);  // Send a raw data capture at 38
        kHz.
144     break;
145
146     case 24:
147       irsend.sendRaw(rawData24, 221, 38);  // Send a raw data capture at 38
        kHz.
148     break;
149
150     case 25:
```

```
151         irsend.sendRaw(rawData25, 221, 38);  // Send a raw data capture at 38
        kHz.
152       break;
153
154       case 26:
155         irsend.sendRaw(rawData26, 221, 38);  // Send a raw data capture at 38
        kHz.
156       break;
157
158       case 27:
159         irsend.sendRaw(rawData27, 221, 38);  // Send a raw data capture at 38
        kHz.
160       break;
161
162       case 28:
163         irsend.sendRaw(rawData28, 221, 38);  // Send a raw data capture at 38
        kHz.
164       break;
165
166       case 29:
167         irsend.sendRaw(rawData29, 221, 38);  // Send a raw data capture at 38
        kHz.
168       break;
169
170       case 30:
171         irsend.sendRaw(rawData30, 221, 38);  // Send a raw data capture at 38
        kHz.
172       break;
173
174       case 31:
175         irsend.sendRaw(rawData31, 221, 38);  // Send a raw data capture at 38
        kHz.
176       break;
177
178       case 32:
179         irsend.sendRaw(rawData32, 221, 38);  // Send a raw data capture at 38
        kHz.
180       break;
181
182       default:
183         if (TempAC > 32) // Limite superior do ar condicionado    32
184           TempAC = 32;
185         if (TempAC < 16) // Limite inferior do ar condicionado    16
186           TempAC = 16;
187     }
188 }
189
```

```
190 // RangeController
191 std::map<String, int> globalRangeValues;
192 // ThermostatController
193 float globalTargetTemp;
194 String globalThermostatMode;
195 bool onPowerState(const String &deviceId, bool &state)
196 {
197   Serial.printf("Device 1 turned %s", state?"on":"off");
198   switch (state)
199   {
200     case 'on':
201         irsend.sendRaw(rawData, 221, 38);  // Envia dados RAW, relacionados
      a ligar o ar condicionado(capture at 38kHz).
202     break;
203 
204     default:
205       irsend.sendRaw(rawData0, 221, 38);
206   }
207   return true; // request handled properly
208 }
209 
210 // RangeController
211 bool onRangeValue(const String &deviceId, const String& instance, int &
      rangeValue) {
212   Serial.printf("[Device: %s]: Value for \"%s\" changed to %d\r\n",
      deviceId.c_str(), instance.c_str(), rangeValue);
213   globalRangeValues[instance] = rangeValue;
214   TempAC = rangeValue;
215   controle();
216   Serial.println(TempAC);
217   return true;
218 }
219 
220 bool onAdjustRangeValue(const String &deviceId, const String& instance, int
      &valueDelta) {
221   globalRangeValues[instance] += valueDelta;
222   Serial.printf("[Device: %s]: Value for \"%s\" changed about %d to %d\r\n"
      , deviceId.c_str(), instance.c_str(), valueDelta, globalRangeValues[
      instance]);
223   globalRangeValues[instance] = valueDelta;
224   return true;
225 }
226 
227 // ThermostatController
228 bool onTargetTemperature(const String &deviceId, float &targetTemp) {
229   Serial.printf("[Device: %s]: Target temperature set to %f\r\n", deviceId.
      c_str(), targetTemp);
```

```
230   globalTargetTemp = targetTemp;
231   TempAC = targetTemp;
232   controle();
233   Serial.println(TempAC);
234   return true; // request handled properly
235 }
236
237 bool onAdjustTargetTemperature(const String &deviceId, float &tempDelta) {
238   globalTargetTemp += tempDelta; // change global target temperature about
       tempDelta
239   Serial.printf("[Device: %s]: Target temperature changed about %f to %f\r\
       n", deviceId.c_str(), tempDelta, globalTargetTemp);
240   tempDelta = globalTargetTemp; // return absolute target temperature
241   return true; // request handled properly
242 }
243
244 bool onThermostatMode(const String& deviceId, String& mode) {
245   Serial.printf("[Device: %s]: Thermostat mode set to %s\r\n", deviceId.
       c_str(), mode.c_str());
246   globalThermostatMode = mode;
247   return true; // request handled properly
248 }
249
250 void handleTemperaturesensor()
251 {
252     // Captura valor da temperatura e Humidade
253     Humidity = dht.readHumidity();                          // captura a
      humidade do ar
254     Temp = dht.readTemperature();                           // captura
      temperatura em  C
255     HITemp = dht.computeHeatIndex(Temp, Humidity, false); // calcula o
      indice de calor em celcius
256     arcondicionado.sendTemperatureEvent(Temp, Humidity);
257     delay(300);
258 }  // if event was sent successfuly, print temperature and humidity to
      serial*/
259
260 // Conex o WiFi
261 void wifiSetup()
262 {
263   //Define o como STA
264   WiFi.mode(WIFI_STA);
265   //Conecta
266   Serial.printf("[WIFI] Conectado ao %s", WIFI_SSID);
267   WiFi.begin(WIFI_SSID, WIFI_PASS);
268   // Espera
269   while (WiFi.status() != WL_CONNECTED){
```

```
270       Serial.print(".");
271       delay(100);
272     }
273     Serial.println();
274     //Conectado
275     Serial.printf("[WIFI] STATION Mode, SSID: %s, IP address: %s\n", WiFi.
          SSID().c_str(),WiFi.localIP().toString().c_str());
276 }
277
278 void setup()
279 {
280     // Inicializa monitor serial
281     Serial.begin(115200);
282
283     // Conex o WiFi
284     wifiSetup();
285
286     // PowerStateController
287     arcondicionado.onPowerState(onPowerState);
288
289     // RangeController
290     arcondicionado.onRangeValue("rangeInstance1", onRangeValue);
291     arcondicionado.onAdjustRangeValue("rangeInstance1", onAdjustRangeValue);
292       // ThermostatController
293     arcondicionado.onTargetTemperature(onTargetTemperature);
294     arcondicionado.onAdjustTargetTemperature(onAdjustTargetTemperature);
295     arcondicionado.onThermostatMode(onThermostatMode);
296
297     SinricPro.onConnected([]{ Serial.printf("[SinricPro]: Connected\r\n"); })
          ;
298     SinricPro.onDisconnected([]{ Serial.printf("[SinricPro]: Disconnected\r\n
          "); });
299     SinricPro.begin(APP_KEY, APP_SECRET);
300
301     Blynk.begin(BLYNK_AUTH_TOKEN, WIFI_SSID, WIFI_PASS);  // Inicializa o
          Blynk
302
303     display.begin(SSD1306_SWITCHCAPVCC, 0x3C);            // Inicializa
          Display OLED com endere o I2C 0x3C
304     display.clearDisplay();
305     display.setTextColor(WHITE);
306     dht.begin();                                  // Inicializa o DHT111
307
308     irsend.begin();                               // Inicializa o envio por
          infravermelho
309
310     timer.setInterval(1000L, myTimer);            // Inicializa o Timer do
```

```
      blynk para enviar os dados para o APP a cada 1s
311
312 }
313
314 void loop()
315 {
316   // Runs all Blynk stuff
317   Blynk.run();
318   timer.run();
319
320   // handle SinricPro commands
321   SinricPro.handle();
322   handleTemperaturesensor();
323
324   // Exibe os valores no Display OLED.
325   display.clearDisplay();               // clear display
326   // Exibe o valor de sensa  o t rmica
327   display.setTextSize(1);               // set text size
328   display.setCursor(0,0);               // set position to display
329   display.print("Sensacao T rmica ");   // set text
330   display.setTextSize(2);
331   display.setCursor(0,10);
332   display.print(HITemp);
333   display.print(" ");
334   display.setTextSize(1);
335   display.cp437(true);                  // Escrever o s mbolo
336   display.write(167);
337   display.setTextSize(2);
338   display.print("C");
339   display.setTextSize(1);               // set text size
340   display.setCursor(0,30);               // set position to display
341   display.print("Temp. Ar-condicionado ");   // set text
342   display.setTextSize(2);
343   display.setCursor(0,40);
344   display.print(TempAC);
345   display.display();                    // display on OLED
346 }
```

Codigos/Projeto_TCC.ino