UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA ELÉTRICA E BIOMÉDICA

Trabalho de Conclusão de Curso

SISTEMA DE AUTOMAÇÃO IOT PARA CONTROLE DE ILUMINAÇÃO E AR-CONDICIONADO

VINICIUS OLIVEIRA PORTILHO

BELÉM
2025

VINICIUS OLIVEIRA PORTILHO

SISTEMA DE AUTOMAÇÃO IOT PARA CONTROLE DE ILUMINAÇÃO E AR-CONDICIONADO

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharias Elétrica e Biomédica da Universidade Federal do Pará como requisito parcial para a obtenção de grau de Bacharel em Engenharia Elétrica.

Orientador: Me. Elen Priscila de Souza Lobato
Coorientador: Profª Drª Maria Emília de Lima Tostes

BELÉM
2025

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD Sistema de Bibliotecas da Universidade Federal do Pará Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

P852s Portilho, Vinicius.
SISTEMA DE AUTOMAÇÃO IOT PARA CONTROLE DE ILUMINAÇÃO E AR CONDICIONADO / Vinicius Portilho. — 2025.
69 f. : il. color.

Orientador(a): Me. Elen Lobato
Coorientação: Profª. Dr.ª Maria Emilia de Lima Tostes Trabalho de Conclusão (Graduação) - Universidade Federal do Pará, Instituto de Tecnologia, Faculdade de Engenharia Elétrica, Belém, 2025.

1. Ar-Condicionado. 2. Automação. 3. Iluminação. 4. Internet das Coisas. I. Título.

CDD 001.642

VINICIUS OLIVEIRA PORTILHO

SISTEMA DE AUTOMAÇÃO IOT PARA CONTROLE DE ILUMINAÇÃO E AR-CONDICIONADO

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharias Elétrica e Biomédica da Universidade Federal do Pará, como requisito parcial para a obtenção de grau de Bacharel em Engenharia Elétrica.

Data de aprovação: 22/09/2025

Conceito: Bom

BANCA EXAMINADORA:

Me. Elen Priscila de Souza Lobato
Orientadora - CEAMAZON/UFPA

Prof.ª Dr.ª Maria Emilia de Lima Tostes
Coorientadora – FEEB/UFPA

Prof. Dr. Wellington da Silva Fonseca
Avaliador Interno - FEEB/UFPA

Prof. Dr. Antonio Roniel Marques de Sousa
Avaliador Externo – SESI/PA

VISTO:

Prof.ª Dr.ª Carminda Célia Moura de Moura Carvalho
Diretora da Faculdade de Engenharia Elétrica e Biomédica

As minhas mães, Wilma Portilho, Valdinea Portilho e Valdisia Portilho, pelo amor, apoio e dedicação incondicionais.

A minha filha, meu bem mais precioso e maior inspiração.

Ao meu tio Valter Portilho (in Memoriam), minha eterna gratidão pelo incentivo que me motivou a ingressar no curso.

AGRADECIMENTOS

Meus agradecimentos a minha família, pela força e pelo incentivo constante em todos os momentos desta jornada. A minha mãe, Valdinea Portilho, e a minha tia, Valdisia Portilho, pelo cuidado, pela presença e pelo apoio de sempre, que fizeram diferença ao longo de todo o caminho. Em especial, a minha tia e madrinha Wilma Oliveira Portilho, que sempre considerei como mãe, pela dedicação, pelo apoio psicológico e pela presença indispensável em minha vida. Sua participação foi fundamental para que eu conseguisse chegar até aqui.

Ao meu tio Valter Oliveira Portilho (in memoriam), que durante muitos anos foi a principal figura paterna em minha vida. Lembro com carinho das vezes em que me incentivava a desmontar e remontar coisas, despertando em mim a curiosidade de entender o que havia por trás de cada funcionamento. Esses momentos marcaram minha infância e foram fundamentais para que eu escolhesse a Engenharia Elétrica como profissão.

A minha orientadora, Me. Elen Priscila de Souza Lobato, sou profundamente grato pela paciência, pelas orientações atenciosas e pela ajuda decisiva no desenvolvimento deste trabalho. A minha coorientadora, Prof.ª Dr.ª Maria Emília de Lima Tostes, registro meu sincero reconhecimento pelas contribuições que enriqueceram a minha formação. Aos membros da minha banca Prof.Dr. Wellingotn Fonseca e aos Prof. Dr. Antonio Roniel Marques, pelas contribuições apontadas para a conclusão deste trabalho.

Ao CEAMAZON, em especial o Laboratório LCADE, pelo suporte, estrutura e oportuni dades oferecidas, que foram essenciais para a concretização deste TCC.

“O importante é não parar de questionar.”

Albert Einstein

RESUMO

Este Trabalho de Conclusão de Curso apresenta o desenvolvimento de um sistema de automação residencial baseado em Internet das Coisas (IoT, do inglês *Internet of Things*) para controle de iluminação e ar-condicionado. Utilizando o microcontrolador ESP32, o sistema integra um servidor HTTP, responsável por hospedar uma interface web acessível via navegador, que permite a comunicação entre a ESP32 e um servidor externo. A solução proporciona controle remoto, em tempo real, dos dispositivos conectados, com baixo custo, fácil implementação e possibilidade de expansão, sendo uma alternativa prática e eficiente para ambientes residenciais ou comerciais.

Palavras-chave: Ar-Condicionado. Automação. Iluminação. Internet das Coisas.

ABSTRACT

This final project presents the development of an Internet of Things (IoT) based home automation system for lighting and air conditioning control. Using the ESP32 microcontroller, the system integrates an HTTP server, responsible for hosting a web interface accessible via a browser, which enables communication between the ESP32 and an external server. The solution provides real time remote control of connected devices, with low cost, easy implementation, and expandability, making it a practical and efficient alternative for residential or commercial environments.

Key-words: Air Conditioning. Automation. Lighting. Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Corrente Alternada (*Alternating Current*)
ADC	Conversor Analógico-Digital (*Analog-to-Digital Converter*)
AJAX	*Asynchronous JavaScript and XML*
API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
C/C++	Linguagens de Programação C e C++
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CSS	*Cascading Style Sheets*
DAC	Conversor Digital-Analógico (*Digital-to-Analog Converter*)
DC	Corrente Contínua (*Direct Current*)
ESP32	Microcontrolador da família *Espressif Systems*
ESP-IDF	*Espressif IoT Development Framework*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
GPIO	*General Purpose Input/Output*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IoT	*Internet of Things* (Internet das Coisas)
IR	*Infrared* (Infravermelho)
ISO/IEC	*International Organization for Standardization / International Electrotechnical Commission*
ITU	*International Telecommunication Union*
LAN	*Local Area Network*
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
LDR	*Light Dependent Resistor* (Resistor Dependente de Luz)
MQTT	*Message Queuing Telemetry Transport*
NIST	*National Institute of Standards and Technology*
NVS	*Non-Volatile Storage*
OASIS	*Organization for the Advancement of Structured Information Standards*

PIR	*Passive Infrared Sensor* (Sensor Infravermelho Passivo)
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
RFID	*Radio-Frequency Identification*
SPI	*Serial Peripheral Interface*
SRAM	*Static Random-Access Memory*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
TLS/SSL	*Transport Layer Security / Secure Sockets Layer*
UART	*Universal Asynchronous Receiver-Transmitter*
USB	*Universal Serial Bus*
VDC / VAC	Voltagem em Corrente Contínua / Corrente Alternada
Wi-Fi	*Wireless Fidelity*
W3C	*World Wide Web Consortium*
WSN	*Wireless Sensor Network*
XML	*Extensible Markup Language*
5G / 4G / 3G	Padrões de rede móvel de quinta, quarta e terceira geração

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

A automação residencial pode ser definida como o conjunto de técnicas que utilizam tecnologias para automatizar funções em uma habitação, sendo também conhecida como domótica. Sua principal motivação está relacionada ao aumento do conforto dos usuários e à execução de tarefas simples de forma automática (COELHO; CRUZ, 2017).

As primeiras iniciativas datam da década de 1980, quando surgiram soluções voltadas para simplificar atividades cotidianas, como a abertura de portões e o acionamento de sistemas de iluminação por meio de controles remotos. Nesse período inicial, também foram introduzidos recursos de segurança, como alarmes. Com a evolução da domótica, consolidaram-se três áreas principais: automação, segurança residencial e conforto (MERÇON, 2022).

Na automação, atuadores são instalados em elementos móveis da edificação, como portões, portas de garagem, venezianas e toldos, muitas vezes associados a sensores que ampliam as funcionalidades. Já a segurança residencial evoluiu para além dos alarmes contra intrusos, abrangendo controle de acesso, monitoramento por câmeras e acionamento automático de luzes para simular a presença de moradores. Por fim, o conforto e a personalização incluem o controle de eletrodomésticos, a regulação da temperatura e o gerenciamento da iluminação (ALVES et al., 2022).

A partir dos anos 2000, surgiu o conceito de casa conectada, que incorpora o uso da internet ao cotidiano, permitindo a integração de múltiplos dispositivos à rede para envio de informações e monitoramento. Essa integração resultou na casa inteligente, marcada pela fusão entre a automação residencial e a conectividade, em que a comunicação e a interatividade entre dispositivos possibilitam ambientes mais dinâmicos e eficientes (JR e FARINELLI, 2018).

Nesse contexto, ganha destaque a Internet das Coisas (IoT, sigla do termo inglês *Internet of Things*), definida como a rede de objetos físicos capazes

de coletar, processar e compartilhar dados pela internet (SOUZA, 2019). Na automação residencial, a IoT amplia as possibilidades de controle e monitoramento em tempo real, conectando desde lâmpadas e eletrodomésticos até sistemas de climatização e segurança (ALVES et al., 2022).

Entre os dispositivos que mais contribuíram para a popularização da automação residencial acessível está o ESP32, um microcontrolador de baixo custo que combina conectividade sem fio e capacidade de controlar diferentes equipamentos. Por conta da sua versatilidade, ele permite criar soluções que vão desde sistemas de iluminação inteligente até o controle de aparelhos como televisores e condicionadores de ar por meio de sinais infravermelhos. Essa praticidade possibilita que usuários registrem e acionem comandos de forma simples, muitas vezes diretamente em uma interface web acessível pelo navegador, tornando a automação mais intuitiva e próxima da rotina cotidiana (SILVA; PEREIRA, 2020).

O conceito de casa inteligente, impulsionado pela IoT e pelo uso de tecnologias como o ESP32, vem se tornando mais acessível, especialmente com os esforços das empresas em simplificar sistemas já existentes. Hoje, é possível encontrar soluções que vão desde geladeiras capazes de enviar alertas quando produtos se esgotam até lâmpadas conectadas a assistentes virtuais, que podem ser controladas por comandos de voz (MARTINS; RIBEIRO, 2021). Esse cenário ganhou ainda mais relevância após a pandemia de Covid-19, quando o home office se tornou parte do cotidiano de muitas pessoas. Nesse contexto, ambientes interligados tecnologicamente passaram a ser vistos como necessários, pois proporcionam mais conforto, praticidade, espaços inteligentes e conectividade sem que os usuários precisem sair de casa (ZANATTA, 2021).

1.2 Motivação

O elevado consumo de energia em sistemas de climatização e iluminação, aliado à demanda crescente por maior comodidade no gerenciamento de dispositivos, motivou a concepção deste projeto. Essa necessidade não se limita apenas ao ambiente residencial, mas também se estende a espaços comerciais, corporativos e institucionais, nos quais a automação pode contribuir significativamente para a redução de custos

operacionais e para o aumento da eficiência. No entanto, a maioria dos sistemas comerciais disponíveis ainda apresenta custos elevados ou requer infraestruturas complexas, o que restringe sua adoção em larga escala. Nesse contexto, o desenvolvimento de uma solução de baixo custo, expansível e de fácil utilização representa um passo importante para democratizar o acesso à automação, tornando a viável em diferentes tipos de edificações e aplicações.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver um sistema de automação baseado em IoT para controle de iluminação e ar-condicionado, visando promover maior conforto, segurança e eficiência energética.

1.3.2 Objetivos Específicos

- Projetar e implementar circuitos eletrônicos utilizando o ESP32 para controle do sistema de iluminação
- Projetar e implementar circuitos eletrônicos utilizando o ESP32 para controle de aparelhos de ar-condicionado via sinal infravermelho.
- Implementar a comunicação entre o ESP32 e o Servidor *Web*, visando escalabilidade e integração com sistemas externos.
- Desenvolver a interface de monitoramento e controle remoto amigável e acessível via dispositivo móvel.

1.4 Estrutura do documento

Este trabalho possui seis capítulos, sendo este o primeiro capítulo introdutório e os outros cinco capítulo estão organizados da seguinte forma:

Capítulo 2 – Trabalhos Correlatos: Apresenta pesquisas e projetos relacionados ao tema, destacando suas contribuições e limitações.

Capítulo 3 – Referencial Teórico: Reúne conceitos e fundamentos necessários para a compreensão do sistema proposto, incluindo tópicos sobre

IoT, automação residencial e as tecnologias utilizadas.

Capítulo 4 – Sistema de Automação IoT para Controle de Iluminação e Ar-Condicionado: Descreve detalhadamente a metodologia adotada, a arquitetura do sistema e os processos de implementação.

Capítulo 5 – Resultados e Discussão: Apresenta os testes realizados, os resultados obtidos e uma análise crítica em relação aos objetivos estabelecidos.

Capítulo 6 – Conclusão: Expõe as considerações finais, as contribuições alcançadas e sugestões para trabalhos futuros.

2 TRABALHOS CORRELATOS

O estudo de trabalhos correlatos é essencial para compreender os avanços já realizados na área de automação residencial baseada em IoT. A análise desses trabalhos permite identificar soluções existentes, limitações técnicas e oportunidades de inovação que justificam a proposta deste trabalho.

2.1 Descrição dos Trabalhos Correlatos

Um dos trabalhos analisados é o de Lobato (2019), intitulado “Desenvolvimento de um Sistema IoT para o controle de iluminação residencial baseado nos princípios da Indústria 4.0”. Nesse estudo, foi implementado um sistema de automação utilizando o ESP32, com comunicação via protocolo MQTT e interação realizada por meio do aplicativo *MQTTDash* para *Android*. A solução foi validada em uma maquete de residência, comprovando a viabilidade do controle remoto da iluminação por dispositivos móveis. Além disso, destacou-se a contribuição do sistema para a eficiência energética e sua aderência aos conceitos da Indústria 4.0.

De forma complementar, Dos Santos et al. (2013) apresentaram o artigo “Automação Residencial com IoT: Aplicações em Ambientes Inteligentes”, no qual propuseram uma solução para controle de cargas elétricas utilizando ESP8266/ESP32, comunicação via MQTT e uma interface web hospedada em servidor externo. Essa abordagem possibilitou a operação das cargas diretamente pelo navegador, ressaltando a flexibilidade da proposta, o baixo custo de implementação e o potencial de expansão para diferentes dispositivos residenciais.

Avançando no uso exclusivo do ESP32, Soares (2023) desenvolveu um protótipo de automação que, além da iluminação, contemplava o controle de portão eletrônico e o monitoramento de sensores de temperatura, umidade e segurança. Toda a operação era realizada por meio de uma interface web própria, que evidenciou o excelente custo-benefício do ESP32 em comparação a outras plataformas. Contudo, o autor apontou fragilidades relacionadas à segurança da interface, aspecto crítico para a confiabilidade do sistema.

Seguindo uma linha semelhante, Américo (s.d.) investigou o desenvolvimento de um sistema de automação residencial de baixo custo, empregando o ESP32 como unidade de controle e sensores acessíveis, como os de temperatura, luminosidade e presença. O objetivo central foi democratizar o acesso a soluções inteligentes, oferecendo alternativas viáveis a famílias brasileiras com recursos limitados. O sistema foi projetado de forma modular, permitindo a adição de novos dispositivos sem grandes alterações na estrutura, e incluiu uma interface simples de monitoramento e acionamento de cargas elétricas em tempo real. Os resultados confirmaram que, mesmo com investimento reduzido, é possível alcançar funcionalidades típicas de sistemas comerciais, embora tenham sido observadas limitações quanto à robustez da interface e à ausência de protocolos de segurança avançados.

Na área de segurança, Silva (2023) propôs um sistema residencial baseado no ESP32, integrando sensores magnéticos em portas e janelas, sensores de movimento (PIR), um buzzer de alarme e notificações automáticas via Telegram. Embora voltado principalmente à proteção patrimonial, o trabalho demonstra a versatilidade do ESP32 ao integrar múltiplos módulos e serviços externos. A utilização do *Telegram* mostrou-se uma solução prática e econômica para envio de alertas em tempo real, dispensando o desenvolvimento de uma interface exclusiva. Os resultados ressaltaram a eficiência na detecção de invasões e a facilidade de uso, mas também revelaram limitações como a dependência de conexão constante com a internet e a falta de redundância em caso de falhas de rede.

Por fim, merece destaque o estudo de caso da Revista FT (2023), que apresentou a implementação de um sistema de automação de baixo custo voltado ao controle de iluminação e monitoramento de cargas elétricas. Utilizando o ESP32 como microcontrolador principal, a proposta incluiu uma interface web hospedada no próprio dispositivo, eliminando a dependência de servidores externos ou aplicativos específicos. Foram utilizados relés como atuadores e sensores de consumo para acompanhamento do gasto energético em tempo real, reforçando a preocupação com eficiência energética. Os resultados confirmaram a viabilidade técnica e a robustez da solução, mesmo em cenários de orçamento reduzido, consolidando o ESP32 como uma

alternativa eficaz para aplicações práticas em ambientes residenciais.

2.2 Comparativo entre este Trabalho e os Correlatos

Enquanto o trabalho de Lobato (2019) concentra-se no controle de iluminação com ESP32, utilizando comunicação via MQTT e interface por meio de um aplicativo *Android*, o estudo de dos Santos et al. (2013) avança ao empregar ESP8266/ESP32 aliado ao mesmo protocolo, oferecendo maior flexibilidade ao criar uma interface web hospedada externamente, que possibilita o controle tanto pelo computador quanto pelo celular. Já Soares (2023) amplia a aplicação do ESP32 para além da iluminação, incluindo o acionamento de portão eletrônico e o monitoramento de sensores ambientais, mas aponta limitações de segurança na interface. De forma semelhante, Américo (s.d.) explora a viabilidade de sistemas modulares e de baixo custo com ESP32 e sensores acessíveis, embora restritos em termos de robustez e proteção dos dados.

No campo da segurança, Silva (2023) direciona o uso do ESP32 para monitoramento residencial, integrando sensores de movimento e magnéticos, buzzer de alarme e envio de notificações via Telegram, demonstrando a versatilidade do microcontrolador, ainda que limitado pela dependência de conexão constante com a internet. Por sua vez, o estudo de caso publicado na Revista FT (2023) reforça o potencial do ESP32 em soluções acessíveis para o controle de iluminação e monitoramento de cargas elétricas, destacando a preocupação com eficiência
energética, mas sem expandir para outras funcionalidades.

O presente Trabalho de Conclusão de Curso, por sua vez, diferencia-se por integrar de forma unificada o controle de iluminação e de ar-condicionado em uma solução baseada exclusivamente no ESP32. Além de manter a comunicação via MQTT, o projeto introduz melhorias significativas ao permitir o registro e a gestão de sinais infravermelhos diretamente em uma interface web simples e intuitiva, eliminando a necessidade de configurações avançadas pelo usuário. Essa abordagem amplia a usabilidade, preserva o baixo custo e mantém o potencial de expansão, consolidando-se como uma alternativa mais completa

e prática para ambientes residenciais.

2.3 Considerações Finais do Capítulo

A análise dos trabalhos correlatos mostra que a automação baseada em IoT, com microcontroladores e protocolos como o MQTT, já oferece soluções viáveis, acessíveis e expansíveis. Este TCC avança nesse cenário ao propor melhorias na forma de implementação do controle de iluminação e ao ampliar o escopo para o ar-condicionado, permitindo o registro e a gestão de sinais infravermelhos por meio de uma interface web simples e intuitiva. Dessa forma, a proposta contribui para aplicações que podem atender tanto ambientes residenciais quanto comerciais e industriais.

3 REFERENCIAL TEÓRICO

Neste capítulo será apresentado o referencial teórico sobre os principais temas e tecnologias utilizadas no desenvolvimento do sistema de automação proposto neste trabalho. Serão abordados os princípios relacionados à IoT, protocolos de comunicação, arquiteturas de automação e tecnologias empregadas, como o microcontrolador ESP32 e a utilização de sinais infravermelhos para controle de dispositivos.

3.1 IoT

3.1.1 Definição

A IoT é compreendida como a interconexão de objetos físicos, através de sensores e atuadores, eletrodomésticos, veículos e máquinas, capazes de identificar-se e comunicar-se entre si por meio de redes de dados. Esses dispositivos coletam, processam e compartilham informações, permitindo o monitoramento e o controle distribuído de processos e ambientes (SOUZA, 2019).

O termo ganhou destaque no início dos anos 2000, sendo definido como uma rede global de objetos conectados, dotados de identidade única e acesso à internet para troca de informações (ASHTON, 2009). Em um sentido mais amplo, IoT representa a convergência de tecnologias de comunicação, sistemas embarcados, computação em nuvem e serviços digitais, possibilitando desde aplicações em cidades inteligentes e indústria 4.0 até soluções acessíveis em automação residencial (ALVES et al., 2022).

A definição proposta pela ISO/IEC 30141:2018 estabelece a IoT como uma infraestrutura de rede que integra pessoas, processos e coisas, fornecendo serviços inteligentes por meio de arquiteturas de referência capazes de garantir interoperabilidade, escalabilidade e segurança (ISO/IEC, 2018). Dessa forma, a IoT ultrapassa a ideia de simples conectividade, tornando-se um ecossistema integrado de coleta e uso de dados em tempo real.

3.1.2 Histórico IoT

O conceito de IoT antecede a popularização do termo. Já nas décadas de 1970 e 1980 existiam estudos voltados para a comunicação entre máquinas (*Marchine-to-Machine* – M2M), especialmente em sistemas de monitoramento remoto e telemetria em indústrias e companhias de energia. Esses sistemas, embora rudimentares em comparação às soluções atuais, já buscavam conectar dispositivos e permitir a transmissão de dados para centrais de controle (SILVA; PEREIRA, 2020).

Nos anos 1990, com o avanço das tecnologias de identificação automática, em especial o RFID (*Radio-Frequency Identification*), surgiram novas possibilidades para o rastreamento de mercadorias e para a integração de processos em cadeias de suprimento. Foi nesse contexto que Kevin Ashton, pesquisador ligado ao *Massachusetts Institute of Technology* (MIT), cunhou o termo Internet of Things em 1999, durante uma apresentação sobre o uso de etiquetas RFID para melhorar a gestão de estoques na empresa Procter & Gamble (ASHTON, 2009). Ashton defendia que, para além de conectar apenas computadores e pessoas, a internet deveria ser capaz de conectar “coisas”, permitindo que os objetos transmitissem informações de forma autônoma, sem depender da ação humana.

No início dos anos 2000, o tema ganhou maior visibilidade quando a *International Telecommunication Union* (ITU) publicou o relatório *The Internet of Things* (2005), no qual destacava a IoT como uma nova fase da internet, caracterizada pela conexão ubíqua de dispositivos físicos. A ITU enfatizou que, além de pessoas e computadores, qualquer objeto cotidiano poderia se tornar parte da rede global, desde eletrodomésticos e veículos até sensores ambientais e equipamentos industriais (ITU, 2005). Essa publicação foi um marco ao consolidar a IoT como um campo de pesquisa e desenvolvimento tecnológico reconhecido internacionalmente.

Durante essa primeira década do século XXI, diversas universidades, laboratórios e empresas começaram a explorar aplicações práticas. Projetos pioneiros incluíam redes de sensores sem fio (*Wireless Sensor Networks – WSNs*) aplicadas à agricultura de precisão, ao monitoramento de desastres naturais e à saúde. Ao mesmo tempo, surgiam iniciativas de automação

residencial, que buscavam conectar sistemas de iluminação, climatização e segurança. Entretanto, a limitada infraestrutura de conectividade e o custo elevado dos dispositivos ainda restringiam a difusão em larga escala (Gubbi et al., 2013).

O crescimento da IoT foi acelerado a partir de 2010, impulsionado por três fatores principais: (i) a massificação da internet banda larga e redes móveis (3G e 4G); (ii) o desenvolvimento de microcontroladores e módulos de baixo custo, como o ESP8266 e posteriormente o ESP32, capazes de integrar conectividade *Wi-Fi* e Bluetooth em um único chip; e (iii) a criação de protocolos de comunicação leves, como o MQTT, projetado para dispositivos com restrições de energia e processamento (OASIS, 2019). Essa combinação abriu espaço para aplicações em domínios até então inviáveis, tornando a IoT acessível também para residências e pequenas empresas.

Na última década, a IoT consolidou-se como um dos pilares da chamada Indústria 4.0, integrando-se a tecnologias como computação em nuvem, Inteligência Artificial e *Big Data*. Essa integração possibilitou não apenas o monitoramento, mas também a análise preditiva e a automação inteligente de processos. Paralelamente, no ambiente residencial, as chamadas casas inteligentes (*smart homes*) tornaram-se realidade com a popularização de assistentes virtuais (como Alexa e *Google Assistant*) e dispositivos de fácil configuração, como lâmpadas, câmeras de segurança e condicionadores de ar conectados (ZANATTA, 2021).

Outro marco importante foi a padronização de arquiteturas e diretrizes pela ISO/IEC 30141:2018, que estabeleceu uma arquitetura de referência para a IoT, destacando princípios de interoperabilidade, escalabilidade, segurança e privacidade. Esse documento fortaleceu a base conceitual e técnica da IoT, orientando a criação de soluções mais confiáveis e sustentáveis (ISO/IEC, 2018).

Atualmente, a IoT está fortemente relacionada a conceitos emergentes, como computação em borda (*edge computing*), computação em névoa (*fog computing*) e 5G, que visam superar limitações de latência, escalabilidade e confiabilidade. Além disso, o tema passou a integrar discussões sobre cibersegurança e privacidade, dado o crescente número de ataques a dispositivos

conectados e a necessidade de normatizar requisitos mínimos de proteção (NIST, 2020).

Assim, o histórico da IoT revela uma evolução contínua: de ideias sobre comunicação entre máquinas e rastreamento de objetos, passando por sua consolidação teórica e experimental no início dos anos 2000, até se tornar hoje uma realidade presente em diversos setores da sociedade. Do ponto de vista acadêmico e tecnológico, a IoT não apenas transformou a forma como interagimos com os objetos ao nosso redor, mas também redefiniu o papel da internet como uma infraestrutura essencial para a vida moderna.

3.2 Microcontrolador ESP32

3.2.1 Definição

O ESP32 é um microcontrolador de baixo custo e alto desempenho desenvolvido pela empresa chinesa *Espressif Systems*, lançado em 2016 como sucessor do popular ESP8266. Assim como seu antecessor, o ESP32 integra conectividade Wi-Fi ao chip, mas adiciona *Bluetooth Classic* e *Bluetooth Low Energy* (BLE), ampliando seu potencial para aplicações em IoT, automação residencial, dispositivos vestíveis e sistemas industriais (ESPRESSIF, 2016).

Além da conectividade, o ESP32 diferencia-se por sua arquitetura baseada em um ou dois núcleos de processamento *Xtensa LX6*, que podem operar a até 240 MHz, com suporte a SRAM interna e opções de memória flash externa. O chip oferece ainda recursos de conversão analógico-digital (ADC), conversão digital-analógica (DAC), interfaces de comunicação serial (UART, SPI, I²C, I²S), sensores de toque capacitivo, PWM, temporizadores e um coprocessador de ultra baixo consumo, que permite operar em modos *deep sleep* adequados para aplicações alimentadas por bateria (WHITE, 2017).

Essa combinação de conectividade, desempenho e eficiência energética tornou o ESP32 uma das plataformas mais utilizadas em projetos acadêmicos e comerciais voltados à IoT. Seu ecossistema de desenvolvimento também contribui para essa popularidade, já que pode ser programado em linguagens como C/C++ por meio da Arduino IDE, do *Espressif*

IoT Development Framework (ESP-IDF) ou ainda com suporte a *MicroPython* (OLIVEIRA; MORAIS, 2020).

3.2.2 Histórico ESP32

O desenvolvimento do ESP32 está diretamente ligado ao sucesso de seu antecessor, o ESP8266, lançado em 2014. Este último revolucionou o mercado de IoT ao oferecer um módulo Wi-Fi de baixo custo e fácil integração, permitindo que dispositivos simples pudessem se conectar à internet. Entretanto, o ESP8266 apresentava algumas limitações, como a ausência de Bluetooth, menor capacidade de processamento e recursos periféricos mais restritos (ESPRESSIF, 2016).

Em resposta às demandas de aplicações mais complexas, a *Espressif* lançou o ESP32 em 2016, agregando novas funcionalidades e maior capacidade computacional. Diferentemente do ESP8266, que contava com apenas um núcleo, o ESP32 foi projetado com arquitetura *dual-core*, permitindo maior paralelismo e eficiência em tarefas que exigem processamento contínuo, como o gerenciamento simultâneo de comunicações sem fio e controle de periféricos (WHITE, 2017).

Outro ponto marcante na evolução do ESP32 foi a ênfase em eficiência energética. A *Espressif* introduziu modos de economia de energia mais avançados, como o deep sleep e o hibernation, que permitem consumo extremamente reduzido durante inatividade, característica essencial para dispositivos móveis e sensores autônomos (ESPRESSIF, 2016).

Desde seu lançamento, o ESP32 tem se consolidado como uma das principais plataformas para prototipagem e desenvolvimento de soluções IoT, sendo amplamente utilizado tanto em contextos acadêmicos quanto em aplicações comerciais. Entre os exemplos de aplicação destacam-se sistemas de automação residencial, monitoramento ambiental, dispositivos vestíveis, controle de drones e até aplicações industriais de maior escala (OLIVEIRA; MORAIS, 2020).

A evolução contínua da família ESP32 também é notável. A *Espressif* lançou variações como o ESP32-S2, o ESP32-C3 (com núcleo RISC-V) e o

ESP32-S3, que incorporam recursos adicionais, como aceleração para Inteligência Artificial, maior segurança criptográfica e suporte a novos padrões de conectividade. Isso mostra a consolidação do ESP32 não apenas como sucessor do ESP8266, mas como uma plataforma em constante expansão, alinhada às demandas crescentes da IoT moderna (ESPRESSIF, 2021).

3.3 Protocolo MQTT

O *Message Queuing Telemetry Transport* (MQTT) é um protocolo de comunicação projetado para sistemas com recursos limitados e conexões de rede instáveis. Criado em 1999 por *Andy Stanford-Clark* (IBM) e *Arlen Nipper* (Arcom), o MQTT foi inicialmente desenvolvido para monitoramento de oleodutos por satélite, onde largura de banda era escassa e a comunicação deveria ser confiável e com baixo consumo energético (BANKS, 2014).

3.3.1 Arquitetura e Funcionamento

O protocolo adota o modelo de comunicação publicador/assinante *(publish/subscribe)*, mediado por um servidor central chamado *broker*. Nesse modelo, dispositivos publicam mensagens em tópicos e outros dispositivos podem assinar esses tópicos para receber as mensagens correspondentes. Essa abordagem desacopla emissores e receptores no tempo, espaço e sincronização, o que aumenta a escalabilidade e a flexibilidade do sistema (OASIS, 2019).

O MQTT opera sobre o TCP/IP, garantindo entrega confiável das mensagens, e possui três níveis de Qualidade de Serviço (QoS):

- QoS 0 (*at most once*): mensagem enviada uma vez, sem confirmação.
- QoS 1 (*at least once*): mensagem confirmada, mas pode haver duplicatas.
- QoS 2 (*exactly once*): garante que cada mensagem chegue apenas uma vez, sendo o mais seguro, porém mais custoso em termos de overhead.

3.3.2 Principais Vantagens

Entre as principais vantagens do MQTT destacam-se:

- Leveza: cabeçalhos pequenos, baixo consumo de banda e processamento.
- Eficiência energética: ideal para dispositivos que operam com bateria, podendo permanecer em sleep mode e reconectar rapidamente.
- Escalabilidade: o modelo *publish/subscribe* permite gerenciar milhares de dispositivos em redes IoT de grande porte.
- *Retained messages e last will*: recursos úteis para manter estados persistentes e lidar com desconexões inesperadas.

Essas características tornaram o protocolo um dos mais utilizados em aplicações de IoT, como automação residencial, cidades inteligentes, indústria 4.0 e monitoramento ambiental (HUNKELER; TRUONG; STANFORD-CLARK, 2008).

3.3.3 Limitações

Apesar de suas vantagens, o MQTT apresenta algumas limitações:

- Depende de uma infraestrutura TCP/IP, o que pode não ser ideal para dispositivos extremamente restritos ou em redes de latência muito alta.
- A simplicidade do protocolo traz desafios de segurança, uma vez que autenticação e criptografia não fazem parte do núcleo do MQTT, devendo ser implementadas via TLS/SSL ou mecanismos externos.
- Em cenários de tempo real rigoroso, o uso do TCP pode introduzir atrasos indesejados em comparação a alternativas baseadas em UDP, como o CoAP (SHELBY et al., 2014).

3.3.4 Aplicações

Atualmente, o MQTT é considerado um dos protocolos padrão para IoT, sendo suportado por diversas plataformas de nuvem (*AWS IoT*, *Microsoft Azure*

IoT Hub, *Google Cloud IoT Core*, *IBM Watson IoT*). Em automação residencial, é amplamente utilizado em sistemas baseados em ESP32 e ESP8266, permitindo controle de iluminação, climatização e eletrodomésticos via interfaces *web* ou assistentes de voz. Sua flexibilidade também se estende a aplicações críticas, como transporte inteligente, telemetria veicular e monitoramento de saúde, onde baixo consumo e confiabilidade são requisitos fundamentais.

3.4 Tecnologias Web

A interface desenvolvida neste trabalho tem como base um conjunto de tecnologias consolidadas no campo do desenvolvimento *web*. A integração desses recursos possibilita a construção de um ambiente interativo, responsivo e capaz de realizar a comunicação em tempo real entre usuários e sistemas embarcados. Nesse sentido, destacam-se o HTML, responsável pela estruturação do conteúdo, o CSS, que define a apresentação visual, o JavaScript, que confere dinamismo e interatividade, e os protocolos HTTP e *WebSocket*, que asseguram a troca eficiente de informações entre cliente e servidor. Na Tabela 1 apresenta-se um resumo das tecnologias *web* utilizadas neste projeto e suas respectivas versões.

Tabela 1 – Técnologias *Web* utilizadas no projeto

Linguagem	**Protocolos**	**Versão**
HTML	*HTTP*	HTML *Living Standard*
CSS	*Websocket*	CSS3 – módulos em atualização contínua
JavaScript		ECMAScript 2025 (edição 16)

Fonte: Autoria Própria

3.4.1 Linguagens

3.4.1.1 HTML

O HTML (*HyperText Markup Language*) é a linguagem de marcação padrão utilizada para estruturar documentos e aplicações na web. Diferente de linguagens de programação, o HTML não executa lógicas de processamento, mas organiza o conteúdo em elementos semânticos, como títulos, parágrafos, botões, tabelas, imagens e formulários (W3C, 2017).

Sua principal função é fornecer ao navegador um conjunto de instruções que define a hierarquia e o significado de cada parte do conteúdo, permitindo que este seja exibido de forma acessível e consistente em diferentes dispositivos. Essa característica garante a interoperabilidade e a universalidade das páginas *web*, pilares fundamentais da internet.

3.4.1.2 CSS

O CSS (*Cascading Style Sheets*) é a linguagem utilizada para definir a camada de apresentação visual de documentos estruturados em HTML ou XML. Sua principal função é separar o conteúdo da forma como ele será exibido, permitindo controlar aspectos como cores, tipografia, espaçamento, bordas, posicionamento e responsividade dos elementos na página (W3C, 2015).

A arquitetura do CSS baseia-se no conceito de "cascata", em que múltiplas regras de estilo podem ser aplicadas a um mesmo elemento, sendo a prioridade definida por critérios de hierarquia, especificidade e ordem de declaração. Essa característica confere flexibilidade e modularidade, possibilitando a criação de interfaces consistentes e de fácil manutenção.

3.4.1.3 *JavaScript*

O *JavaScript* é uma linguagem de programação interpretada, orientada a objetos e amplamente utilizada no desenvolvimento web para proporcionar dinamismo e interatividade às páginas. Diferentemente do HTML, que estrutura o

conteúdo, e do CSS, que define a apresentação visual, o JavaScript atua na camada de comportamento, permitindo a manipulação de elementos da interface em tempo real de acordo com as ações do usuário (FLANAGAN, 2013).

Seu funcionamento está diretamente ligado ao *Document Object Model* (DOM), uma representação hierárquica dos elementos de uma página. Através da manipulação do DOM, o *JavaScript* possibilita alterações dinâmicas, como modificar textos, ocultar ou exibir botões, atualizar tabelas e validar formulários, sem a necessidade de recarregar a página. Além disso, recursos modernos como AJAX e APIs de comunicação assíncrona ampliam seu potencial, permitindo que a aplicação troque dados com o servidor em segundo plano.

3.4.2 Protocolos

3.4.2.1 HTTP

O HTTP (*Hypertext Transfer Protocol*) é o protocolo de comunicação fundamental da *World Wide Web*, responsável por padronizar a transferência de informações entre clientes (navegadores) e servidores. Baseado em um modelo de requisição-resposta, o HTTP define como mensagens devem ser estruturadas, enviadas e interpretadas, possibilitando o acesso a recursos como páginas HTML, folhas de estilo CSS, scripts *JavaScript* e arquivos multimídia (FIELDING; RESCHKE, 2014).

Cada transação HTTP ocorre de forma independente: o cliente envia uma requisição contendo um método (como *GET*, *POST*, *PUT* ou *DELETE*), um cabeçalho com metadados e, opcionalmente, um corpo de mensagem. O servidor, por sua vez, processa essa requisição e devolve uma resposta, geralmente acompanhada de um código de status que indica o sucesso ou falha da operação. Essa simplicidade e clareza de funcionamento contribuíram para a consolidação do HTTP como a espinha dorsal da web moderna.

3.4.2.2 *WebSockets*

O *WebSocket* é um protocolo de comunicação bidirecional que permite a

troca contínua de dados entre cliente e servidor por meio de uma única conexão persistente. Diferentemente do modelo tradicional do HTTP, baseado em múltiplas requisições independentes, o WebSocket estabelece um canal único em que ambas as partes podem enviar e receber informações em tempo real, sem a necessidade de abrir novas conexões (FETTE; MELNIK, 2011).

O funcionamento do *WebSocket* inicia-se com um processo denominado handshake, em que uma requisição HTTP especial é enviada pelo cliente solicitando a atualização do protocolo para *Web Socket*. Uma vez aceita pelo servidor, a conexão é estabelecida e permanece aberta até que uma das partes a encerre. Essa abordagem reduz a latência, o tráfego de rede e o consumo de recursos, tornando o protocolo particularmente adequado para aplicações que demandam respostas imediatas.

3.5 Considerações Finais do Capítulo

O referencial teórico apresentado permitiu contextualizar os principais conceitos e tecnologias que fundamentam o sistema de automação proposto. A IoT foi abordada como um ecossistema em expansão, apoiado em padrões de interoperabilidade, enquanto o ESP32 destacou-se por sua versatilidade, baixo custo e recursos de conectividade. Da mesma forma, o protocolo MQTT foi identificado como solução eficiente para comunicação em ambientes com restrições, e os sinais infravermelhos mostraram-se relevantes para ampliar a integração com dispositivos tradicionais, como sistemas de climatização.

No âmbito da interface, foram discutidas as tecnologias web, incluindo HTML, CSS, *JavaScript*, HTTP e *WebSocket*. Em conjunto, essas ferramentas viabilizam páginas responsivas, interativas e capazes de operar em tempo real, atendendo às demandas de usabilidade e eficiência do sistema. Assim, os elementos apresentados neste capítulo fornecem a base conceitual e técnica necessária para o desenvolvimento e implementação da solução proposta, garantindo sua aplicabilidade em diferentes contextos de automação.

4 SISTEMA DE AUTOMAÇÃO IOT PARA CONTROLE DE ILUMINAÇÃO E AR-CONDICIONADO

Neste capítulo será apresentado o sistema desenvolvido. Serão descritas as tecnologias e materiais utilizados, os *scripts* com os códigos usados para programar o microcontrolador ESP32, e os protótipos dos módulos do sistema IoT criados.

4.1 Materiais Utilizados

Foram desenvolvidos dois protótipos, sendo um para controle de iluminação e outro para controle de ar-condicionado. Em ambos os casos, os dois protótipos utilizam o módulo de desenvolvimento ESP32 DEV KIT V1, que funciona como o dispositivo central do sistema. Esse microcontrolador é responsável por armazenar e processar os dados, além de realizar a conexão e comunicação entre todos os componentes que compõem a solução. A escolha desse módulo se justifica por sua versatilidade, pois ele disponibiliza interfaces de comunicação, *Wi-Fi*, além de apresentar baixo consumo de energia e custo acessível em relação às funcionalidades que oferece. A Figura 1 mostra o módulo ESP32 DEV KIT V1 utilizado no projeto.

Figura 1 – Módulo ESP32 DEV KIT V1

Fonte: (OLIVEIRA, 2019)

Produzido pela empresa DOIT, o módulo apresenta diversas vantagens,

entre elas o fato de ser uma plataforma de *hardware* livre e poder ser programado em linguagem C/C++, que é amplamente utilizada e de código aberto. Outra característica importante é a possibilidade de ser programado por meio da IDE do Arduino, recurso que simplifica o processo de desenvolvimento. Além disso, conta com uma porta USB, que facilita tanto a alimentação quanto o manuseio do dispositivo. O Anexo A apresenta o PinOut do módulo ESP32 DEV KIT V1, enquanto a Figura 2 mostra o chip ESP-WROOM-32, que compõe este módulo.

Figura 2 – Chip ESP32-WROOM-32

Fonte: (ESPRESSIF, 2019)

A Tabela 2, apresenta as especificações do chip ESP 32 versão ESP-WROOM-32.

Tabela 2 – Especificações ESP-WROOM-32

Categoria	Descrição
Certificações	**RF:** FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC **Wi-Fi:** Aliança Wi-Fi **Bluetooth:** BQB **Ecológica:** RoHS/REACH
Wi-Fi	**Protocolos:** 802.11 b/g/n (802.11n até 150 Mbps), A-MPDU e A-MSDU **Faixa de frequência:** 2,4 GHz a 2,5 GHz
Bluetooth	**Protocolos:** Bluetooth v4.2 BR/EDR e BLE specification **Rádio:** Receptor NZIF com sensibilidade de –97 dB; transmissor Classe 1, 2 e 3 – AFH **Áudio:** CVSD e SBC
Hardware	**Interfaces do módulo:** Cartão SD, UART, SPI, SDIO, I²C, LED PWM, Motor PWM, I²S, IR, contador de pulsos, GPIO, sensor de toque capacitivo, ADC, DAC **Sensor integrado:** Sensor Hall (magnético) **Cristal integrado:** Cristal de 40 MHz **Flash SPI integrado:** Padrão 4 MB (pode variar) **Memória interna ROM:** 448 KB **Memória interna SRAM:** 520 KB **Tensão de operação:** 2,7 V a 3,6 V **Tensão recomendada:** 3,3 V **Dimensões do chip:** (18,00 ± 0,10) mm × (25,50 ± 0,10) mm × (3,10 ± 0,10) mm **Dimensões do módulo ESP32 DEV KIT V1:** 27,5 × 51,0 mm

Fonte: (ESPRESSIF, 2019) – Adaptado.

De acordo com o *Datasheet* da fabricante do chip ESP32 (Espressif, 2019), ele contém um sensor de temperatura interno, tem o total de 36 GPIOS, dos quais 2 são Conversores Digital-Analógico (DAC),18 pinos são Conversores

Analógico-Digital (ADC), 10 podem funcionar como sensor de toque capacitivo e os pinos de tensão, sendo eles:

- Vin (5V) – Pino de saída que fornece 5V;
- 3.3V – Pino de saída que fornece 3.3V e uma corrente máxima de 80 mA;
- GND – Pino mais conhecido como “Terra”.

Outro componente presente em ambos os protótipos é a fonte de alimentação Hi-Link. Para o módulo desenvolvido, utilizou-se uma minifonte Hi-Link, responsável por converter a tensão de entrada alternada, que pode variar entre 100 e 240 V (AC), em uma tensão contínua estabilizada de 5 V (DC) na saída. A Figura 3 apresenta a fonte de energia Hi-Link empregada no projeto.

Figura 3 – Fonte de Energia Hi-Link.

Fonte: (HLKTECH, 2023)

A Tabela 3 , apresenta as especificações da fonte *Hi-Link*.

Tabela 3 – Especificações Hi-Link

Parâmetro	Valor
Tensão de entrada	100–240 VAC
Corrente máxima de entrada	0,1 A
Frequência	50–60 Hz
Tensão de saída	5 VDC
Corrente máxima de saída	0,6 A
Potência	3 W

Fonte: Autoria Própria

4.1.1 Materiais Utilizado no Protótipo de Controle de Iluminação

No desenvolvimento do circuito destinado ao controle de iluminação foram empregados os seguintes componentes:

• ESP32: responsável pelo processamento das informações e pela conectividade do sistema.

• Módulo relé de duas vias: utilizado para realizar a comutação da carga, possibilitando o acionamento e desligamento da lâmpada.

• Fonte de alimentação Hi-Link 5V / 0,6 A: responsável por fornecer a energia necessária para o funcionamento do conjunto.

• Interruptor Comum

Os materiais empregados na construção do protótipo de iluminação, juntamente com seus respectivos preços, estão apresentados na Tabela 4, que também evidencia o custo total do sistema. Ressalta-se que, para esse levantamento, foram considerados apenas os componentes diretamente relacionados ao protótipo em si, desconsiderando itens auxiliares utilizados nos testes, como a lâmpada de carga e o interruptor simples.

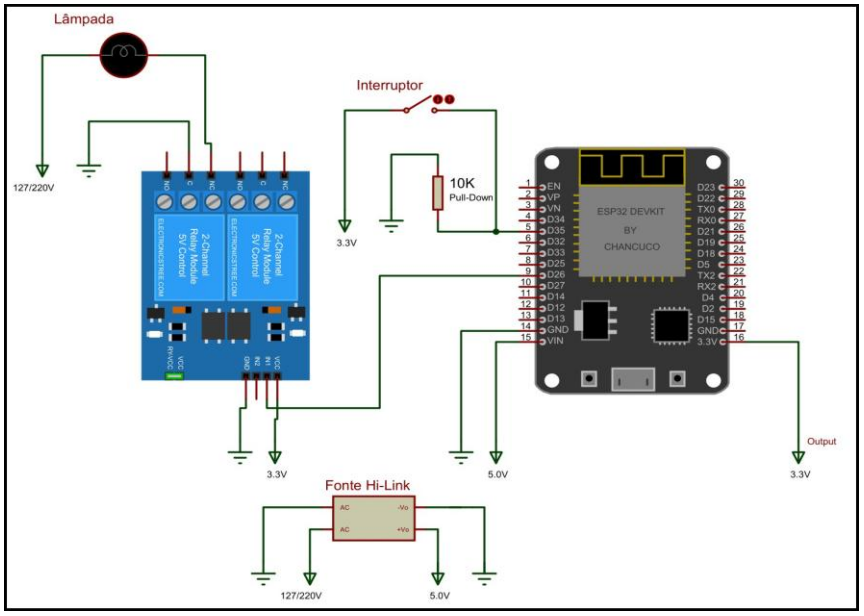
Tabela 4 – Custo do Protótipo do Controle de Iluminação

Componentes	**Valor (R$)**
Módulo ESP32 DEV KIT V1	R$ 56,90
Módulo relé (2 canais)	R$ 12,26
Resistor pull-down 10 kΩ	R$ 0,10
Fonte Hi-Link	R$ 19,72
Valor total	**R$ 88,98**

Fonte: Autoria Própria

Na Figura 4 é possível visualizar o esquemático do circuito integrado.

Figura 4 – Esquemático do Circuito do Controle de Iluminação



Fonte: CHANCUCO - Adaptado

No circuito de iluminação, a lâmpada é acionada pelo relé eletromecânico, que por sua vez é controlado pelo ESP32. O microcontrolador interpreta se o comando de acionamento foi originado do interruptor simples, em modo manual, ou de uma comutação remota via internet, realizada pela interface web. Para garantir confiabilidade no funcionamento, foi empregado um resistor de *pull-down* no circuito de entrada do interruptor, evitando a ocorrência de sinais fantasmas ou estados indefinidos que poderiam provocar acionamentos indevidos da carga.

4.1.1.1 Módulo Relé de Duas Vias

Foi utilizado um módulo relé de dois canais, totalizando dois relés disponíveis para o controle das cargas. Esses relés funcionam como atuadores do sistema, recebendo os comandos de acionamento responsáveis por ligar e desligar as lâmpadas. A Figura 5 apresenta o módulo sólido com dois canais empregado no protótipo.

Figura 5 – Módulo Relé com dois canais

Fonte: MERCADO LIVRE,2025

Esse módulo possui o total de 4 pinos, sendo eles 2 dois pinos de controle (CH1 e CH2), um pino de alimentação (DC+) e o pino GND (DC-). Esse tipo de relé é ligado estado lógico baixo (*LOW*), com tensão entre 0 e 2,5V e desligado quando é aplicada uma tensão de 3 a 5V nos pinos CH1 e CH2. De acordo com ROBOCORE, esse módulo possui as seguintes especificações da Tabela 5.

Tabela 5 – Específicação Módulo Relé

Parâmetro	**Valor**
Tensão de operação	5 VDC
Tensão máxima de carga	250 VAC / 30 VDC
Corrente máxima de carga	10 A
Quantidade de canais	2
Pinos	CH1, CH2, DC+ e DC–
Dimensões	5 × 3,9 × 1,7 cm
Peso	28 g

Fonte: ROBOCORE – Adaptado.

4.1.2 Materiais Utilizados no Protótipo do Controle do Ar-condicionado

Para a implementação do circuito destinado ao controle do ar-condicionado foram empregados os seguintes componentes:

• ESP32: responsável pelo processamento das informações e

pela conectividade do sistema.

• LED emissor infravermelho (IR): componente responsável por transmitir os sinais de controle ao ar-condicionado.

• Receptor infravermelho (IR): utilizado para captar os sinais provenientes do controle remoto original, possibilitando o registro e a reprodução dos comandos.

• Fonte de alimentação Hi-Link 5V / 0,6 A: responsável por fornecer a energia necessária para o funcionamento do conjunto.

Os materiais empregados na construção do protótipo de iluminação, juntamente com seus respectivos preços, estão apresentados na Tabela 6, que também evidencia o custo total do sistema.

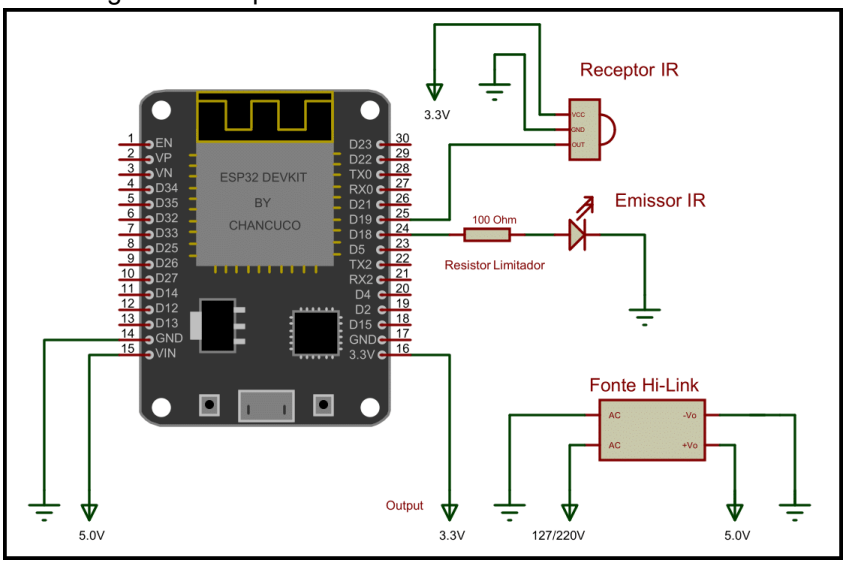
Tabela 6 – Custo do Protótipo do Controle do Ar-Condicionado

Componentes	**Valor (R$)**
Módulo ESP32 DEV KIT V1	R$ 56,90
Emissor IR TSAL6102	R$ 1,50
Receptor IR TSOP4830	R$ 9,00
Resistor limitador 100 Ω	R$ 0,10
Fonte Hi-Link	R$ 19,72
Valor total	**R$ 87,22**

Fonte: Autoria Própria

Na Figura 6 é possível visualizar o esquemático do circuito integrado.

Figura 6 – Esquemático do Circuito do Ar-Condicionado



Fonte: CHANCUCO – Adaptado

4.1.2.1 LED Emissor Infravermelho

O TSAL6124 é um LED emissor de infravermelho (díodo emissor de IR), fabricado pela VISHAY. Trata-se de um componente do tipo *through-hole* (furo passante), encapsulado em pacote de 5 mm (T-1 3/4), com lente transparente. Conforme presente na Tabela 7 é projetado para emissão na faixa de 940 nm, que corresponde ao comprimento de onda tipicamente utilizado em circuitos de controle por infravermelho, como os empregados em aparelhos de ar-condicionado (VISHAY, 2025).

Tabela 7 – LED Infravermelho

Parâmetro	Valor
Tensão de funcionamento	0,8 V – 1,5 V
Corrente de funcionamento	20 mA
Comprimento de onda emitido	940 nm

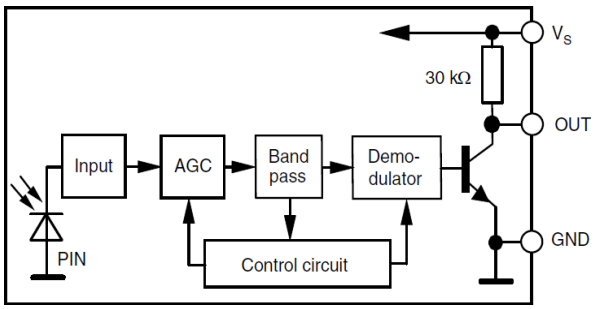
Fonte: VISHAY – Adaptado

4.1.2.2 Receptor IR

O TSOP4838 é um receptor infravermelho integrado, amplamente utilizado em sistemas de controle remoto. Ele é projetado especificamente para detectar sinais de luz infravermelha modulada em 38 kHz, que é a frequência mais comum em dispositivos eletrônicos como televisores, sistemas de áudio e aparelhos de ar-condicionado.

Internamente, como mostra a Figura 7 o componente possui um fotodiodo sensível à radiação na faixa de 940 nm, associado a um pré-amplificador e a um filtro eletrônico sintonizado para 38 kHz. Essa combinação permite que o TSOP4838 rejeite fontes de luz ambiente, como iluminação fluorescente ou luz solar, e responda apenas aos sinais modulados corretamente.

Figura 7 – Circuito Interno do Receptor IR



Fonte: (VISHAY, 2025)

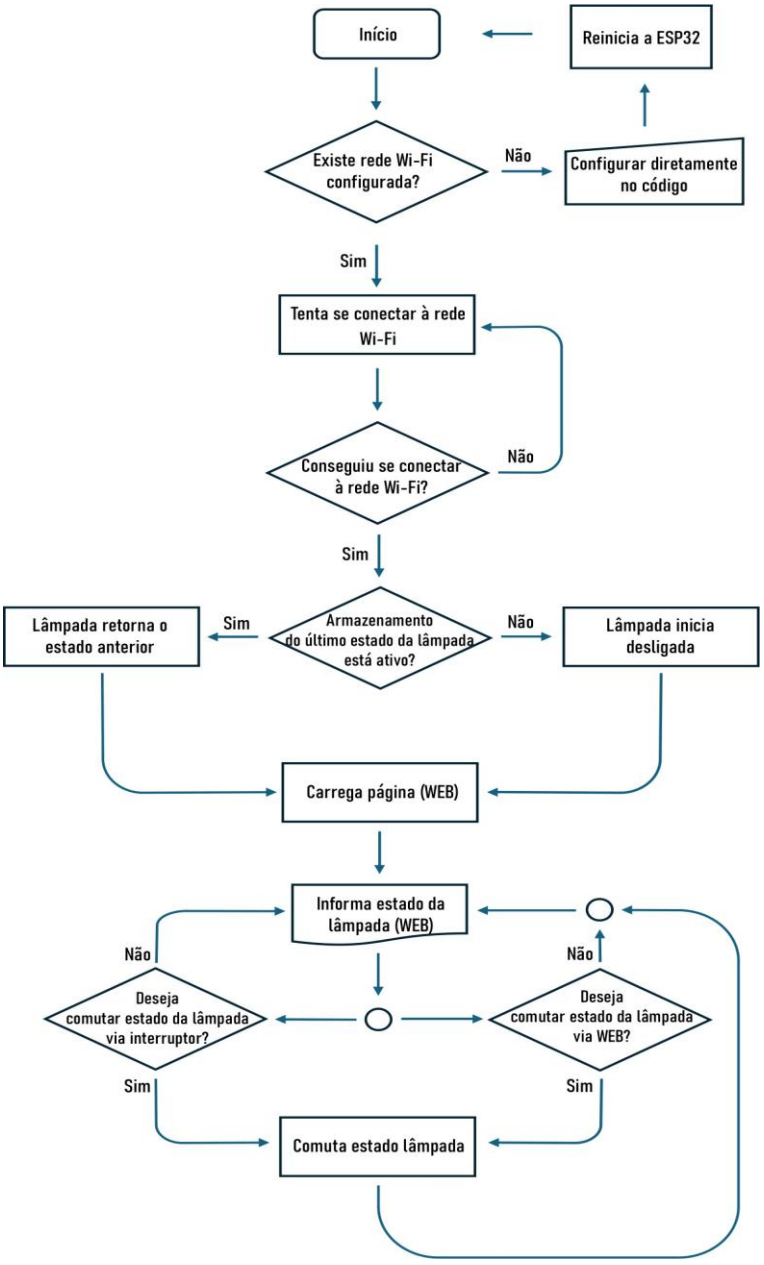
O sinal de saída do receptor é digital: nível baixo (0) quando detecta a presença de pulsos infravermelhos e nível alto (1) quando não há detecção. Essa característica facilita a interface direta com microcontroladores, como o ESP32, permitindo que os dados recebidos sejam processados por software e interpretados como comandos de controle remoto.

4.2 Lógica de funcionamento

4.2.1 Lógica de Funcionamento do Circuito de Iluminação

O processo tem início com a verificação da existência de uma rede Wi-Fi previamente configurada no dispositivo. Se nenhuma rede estiver registrada, a configuração deverá ser realizada diretamente no código. Caso exista uma rede salva, o ESP32 tenta se conectar; se a conexão falhar, o processo retorna à etapa de tentativa, garantindo persistência. Uma vez estabelecida a conexão, o sistema verifica se a função de armazenamento do último estado da lâmpada está habilitada. Se essa função estiver ativa, a lâmpada retoma automaticamente o estado anterior em que se encontrava antes da reinicialização ou queda de energia. Caso contrário, ela inicia desligada, assegurando previsibilidade e segurança no funcionamento. Esse processo é demonstrado através do diagrama de fluxo na Figura 8.

Figura 8 – Diagrama do Código do Circuito de Iluminação

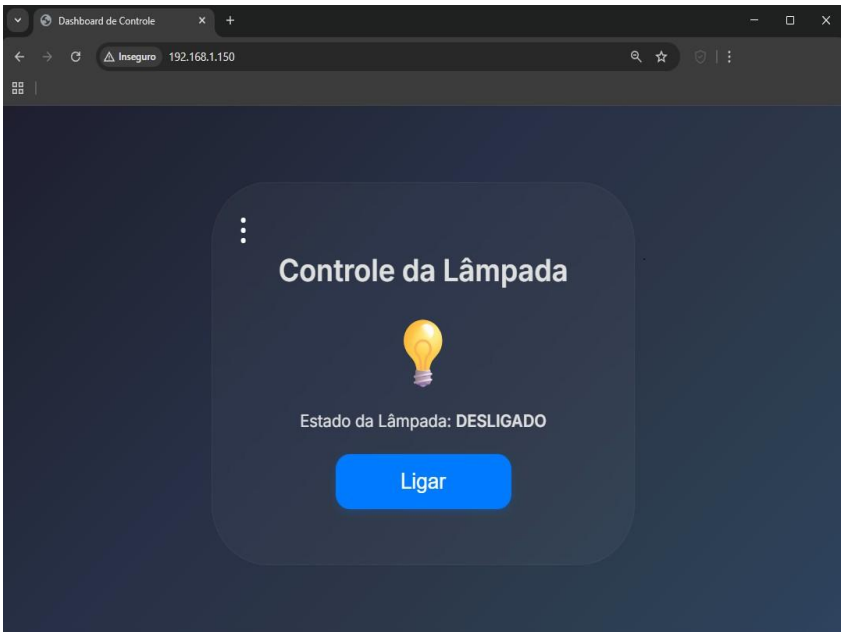


Fonte: Autoria Própria

O acionamento da lâmpada pode ocorrer de duas formas distintas e complementares. A primeira é por meio do interruptor físico, conectado a uma entrada digital do ESP32. Para assegurar a confiabilidade da leitura, foi utilizado um resistor de *pull-down* de 10 kΩ, responsável por manter o pino em nível lógico baixo quando o interruptor está aberto. Essa configuração evita que o microcontrolador registre estados indefinidos ou oscilações causadas por ruídos elétricos. O valor de 10 kΩ foi escolhido por ser amplamente utilizado em aplicações digitais como resistor de pull-down. (ESPRESSIF SYSTEMS, 2022)

A segunda forma de acionamento é através da interface web hospedada no próprio ESP32, acessível por qualquer navegador conectado à mesma rede, como mostra a Figura 9. Essa página possibilita ao usuário interagir remotamente com o sistema, enviando comandos em tempo real por meio de protocolos como HTTP ou *WebSocket*.

Figura 9 – Controle da Lâmpada Via Web



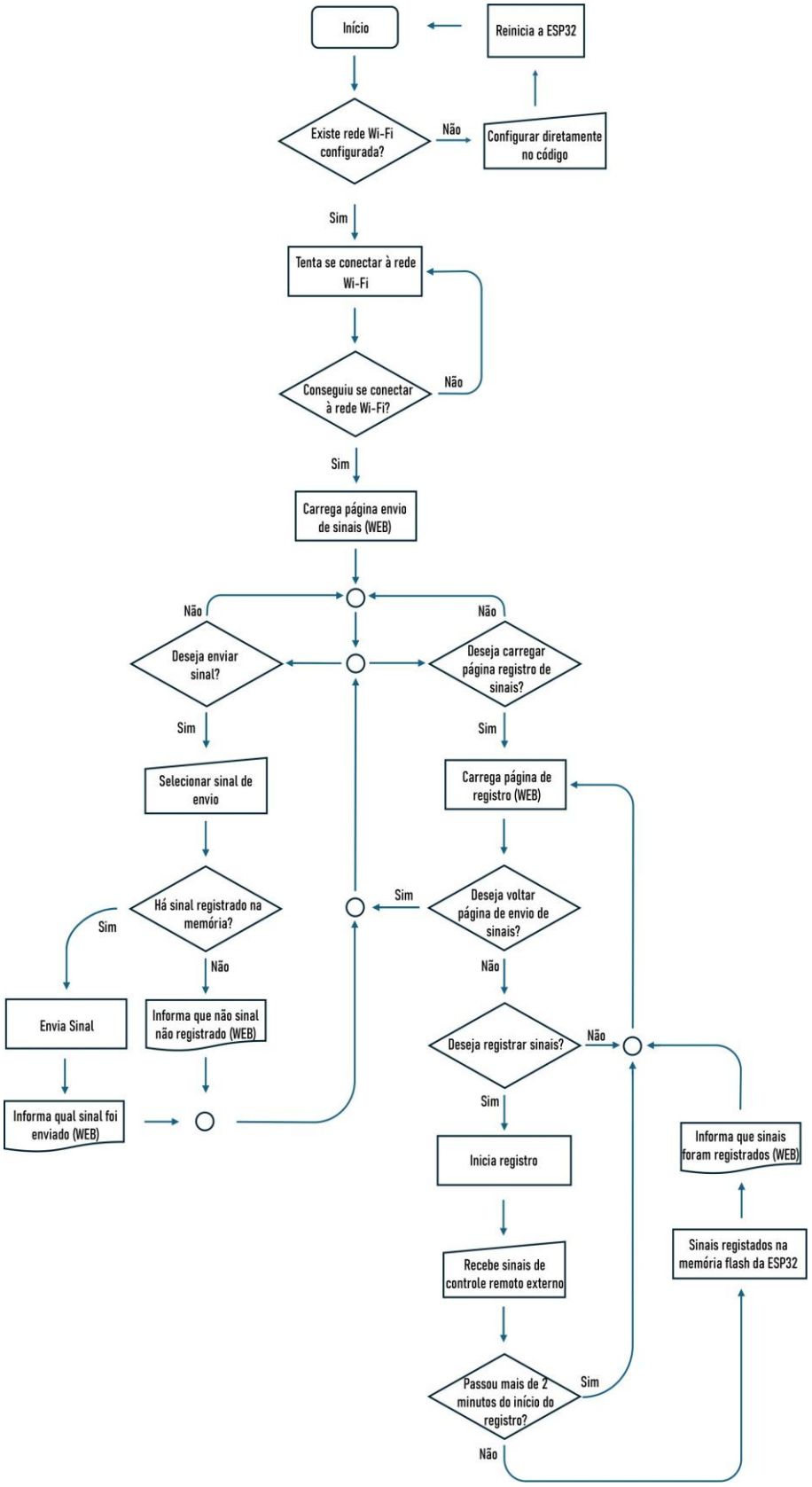
Fonte: Autoria Própria

Além disso, o sistema pode ser configurado para salvar o último estado da lâmpada em memória não volátil (EEPROM). Essa configuração é realizada diretamente pela interface web hospedada no ESP32, onde o usuário pode habilitar ou desabilitar a função de preservação do estado. Com isso, após uma queda de energia ou reinicialização do microcontrolador, a lâmpada retorna ao mesmo estado em que se encontrava anteriormente.

4.2.2 Lógica de Funcionamento do Circuito de Ar-Condicionado

O funcionamento do sistema de controle do ar-condicionado pode ser melhor compreendido a partir do fluxograma apresentado na Figura 10. Esse diagrama ilustra, de forma sequencial e lógica, as etapas percorridas pelo ESP32 desde a inicialização até a execução das funções principais do protótipo.

Figura 10 – Diagrama Principal do Código do Circuito do Ar-Condicionado

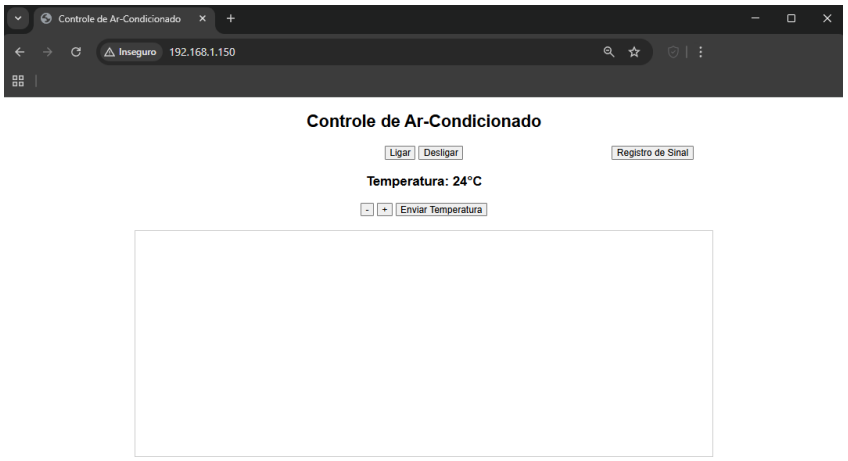


Fonte: Autoria Própria

O funcionamento do sistema do ar-condicionado inicia com a verificação da rede Wi-Fi. Caso não exista uma rede previamente configurada, torna-se necessário definir os parâmetros de conexão diretamente no código. Se a rede já estiver configurada, o ESP32 tenta estabelecer a conexão. Na hipótese de falha, o processo de tentativa é repetido até que a conexão seja concluída com sucesso.

Uma vez conectado à rede Wi-Fi, a página inicial do sistema é carregada, permitindo a interação do usuário com as funções disponíveis, conforme mostra a Figura 11. A partir desse ponto, duas operações principais podem ser realizadas: o envio de sinais ou o registro de novos sinais.

Figura 11 – Página Inicial do Controle do Ar-Condicionado

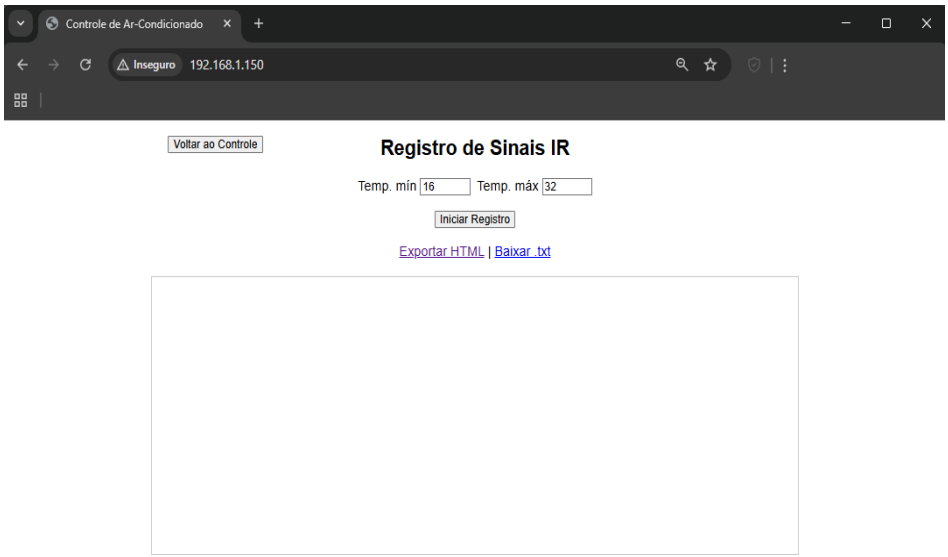


Fonte: Autoria Própria

Ao solicitar o envio de um comando, o sistema consulta a memória do ESP32 para verificar a existência do sinal correspondente. Se o sinal estiver previamente registrado, ele é transmitido e o usuário é notificado por meio da mensagem “Sinal enviado”. Caso contrário, o sistema retorna a mensagem informando que o sinal não se encontra registrado.

Para realizar o registro dos sinais do controle remoto do ar-condicionado, o usuário deve clicar no botão “Registro de Sinal”, o que o direcionará para uma nova página destinada especificamente ao processo de registro dos sinais. Conforme mostra a Figura 12

Figura 12 – Página Registro de Sinais



Fonte: Autoria Própria

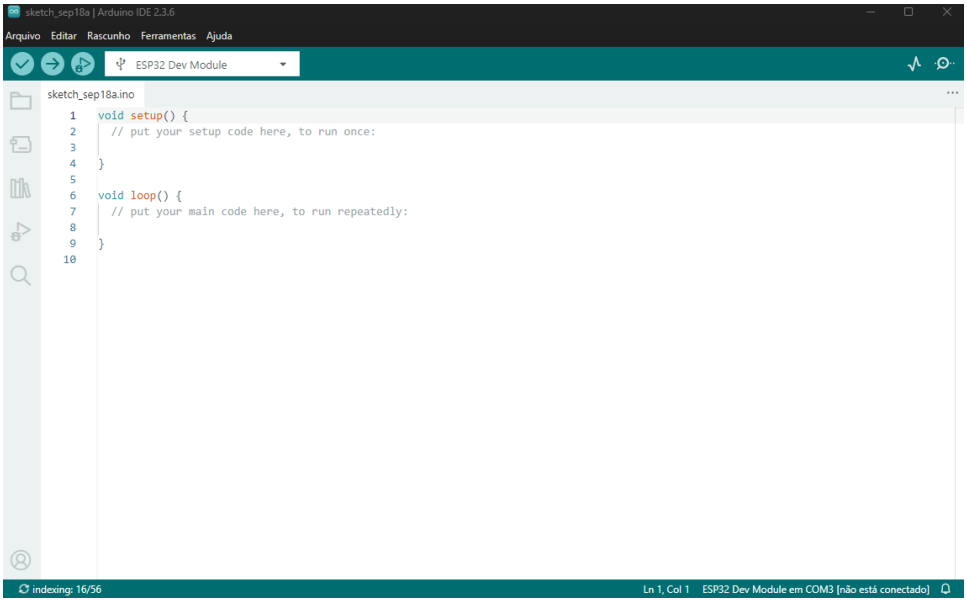
No processo de registro, ao iniciar o procedimento, o usuário dispõe de até dois minutos para concluí-lo. Caso esse tempo seja excedido, o sistema retorna automaticamente para a página inicial do registro. Os sinais devidamente registrados dentro do prazo são gravados de forma permanente na memória flash do ESP32, o que garante a preservação das informações mesmo após desligamentos ou reinicializações do dispositivo.

4.3 Ambiente de Desenvolvimento Integrado do Arduino

Embora a própria fabricante *Espressif* disponibilize o ambiente de desenvolvimento oficial denominado ESP-IDF (*Espressif IoT Development Framework*), neste trabalho optou-se pela utilização da IDE do Arduino, por se tratar de uma ferramenta mais didática, simples e de fácil manuseio para programar o módulo ESP32. O Arduino IDE é um software livre desenvolvido originalmente para os módulos Arduino, utilizando a linguagem C/C++. Nele, os programas recebem o nome de *sketch* e são estruturados em duas funções principais: setup(), executada uma única vez no início do programa, responsável pelas configurações iniciais (como a definição do modo dos pinos e a inicialização de bibliotecas), e loop(), que é executada continuamente, representando o núcleo lógico do programa (ARDUINO, 2019). A Figura 13 apresenta a interface principal da IDE do Arduino, exibindo um *sketch* com suas

funções básicas.

Figura 13 – Janela Principal do IDE Arduino



Fonte: Autoria Própria

Existem alguns pré-requisitos para que o módulo ESP32 seja corretamente reconhecido na IDE do Arduino, sendo eles:

1º Pré-requisito: é necessário instalar previamente o driver responsável pela comunicação entre o computador e o módulo. Nos ESP32, essa comunicação é feita por meio de um conversor USB–UART integrado à placa. Os dois circuitos integrados (CIs) mais comuns utilizados para essa função são:

CP210x (Silicon Labs): exige a instalação do driver “CP210x USB to UART Bridge”, que permite ao sistema operacional reconhecer o dispositivo na porta serial.

CH340/CH341 (WCH): utilizado em alguns módulos alternativos de ESP32, necessitando do driver correspondente “*CH340 USB to UART Driver*”.

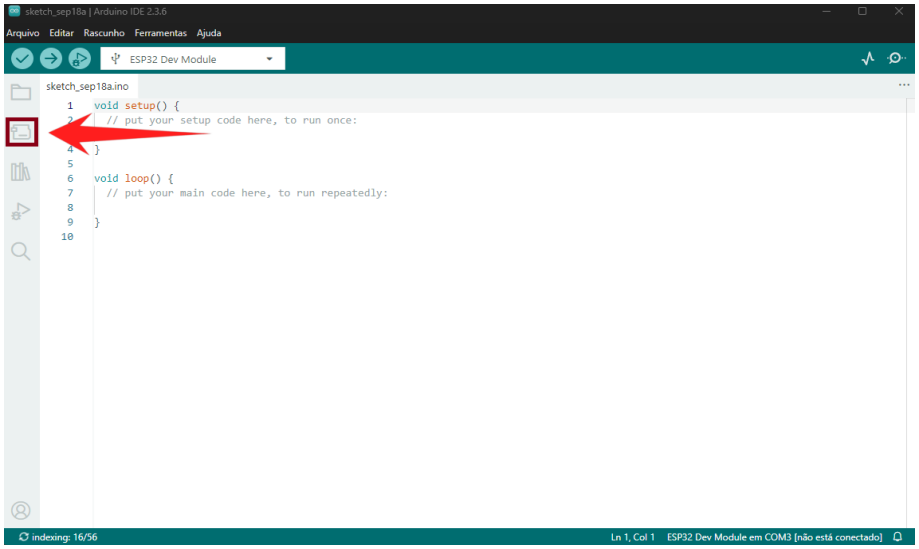
O módulo ESP32 DEVKIT V1, produzido pela DOIT, utiliza como interface de comunicação o conversor USB–UART CP210x, da *Silicon Labs*.

2º Pré-requisito: o instalador (.exe) da IDE do Arduino deve ser baixado diretamente do site oficial do Arduino, e não da loja do sistema operacional, pois apenas a versão oficial garante compatibilidade completa com as bibliotecas e placas adicionais.

Mediante a esses pré-requisitos, basta executar os seguintes passos na IDE do Arduino para instalar o módulo ESP32:

1º Passo: Abra a IDE do Arduino e clique na barra lateral esquerda no ícone “Gerenciador de Placas”, destacado em vermelho. Conforme mostra a Figura 14.

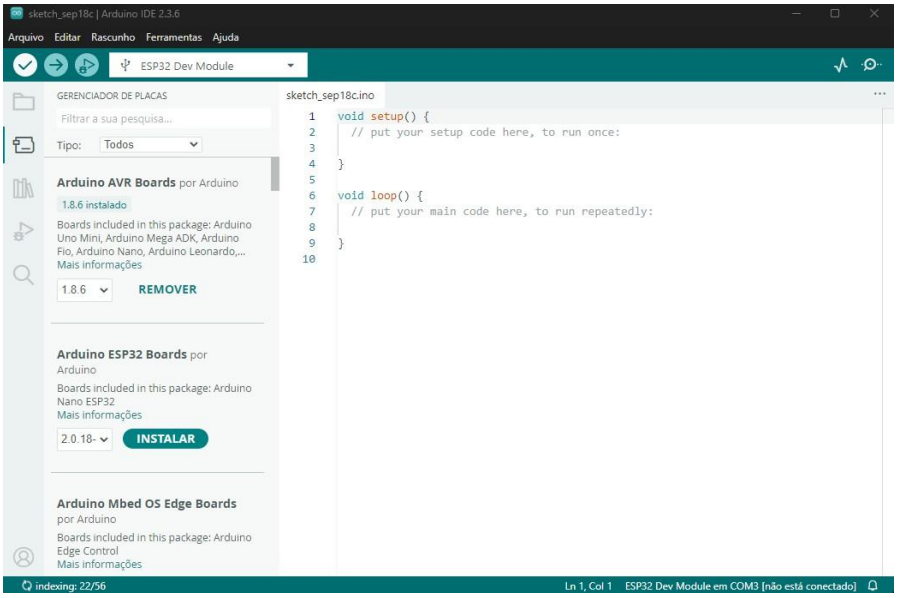
Figura 14 – Janela Principal do IDE Arduino



Fonte: Autoria Própria

2º Passo: Na barra de pesquisa do gerenciador, digite “ESP32” para filtrar os pacotes disponíveis. Conforme mostra a Figura 15.

Figura 15 – Segundo Passo de Instalação do Módulo ESP32 no IDE do Arduino



Fonte: Autoria Própria

3º Passo: Localize a opção “ESP32 por *Espressif Systems*” e clique em Instalar

4.4 Análise do Código

Esta seção tem como objetivo apresentar e discutir os principais aspectos dos códigos desenvolvidos para os protótipos, destacando as bibliotecas utilizadas, a organização das rotinas implementadas e alguns dos desafios enfrentados durante o processo de elaboração. A análise busca evidenciar tanto os recursos empregados na construção das funcionalidades quanto as soluções adotadas para superar limitações técnicas e garantir o funcionamento adequado do sistema. O código completo do projeto está disponível em: <https://gitlab.com/vinicius.oliveira.portilho-group/vinicius.oliveira.portilho-project/-/tree/770ebfdc5cc80cb91135997db82d998f94a1a34e/>.

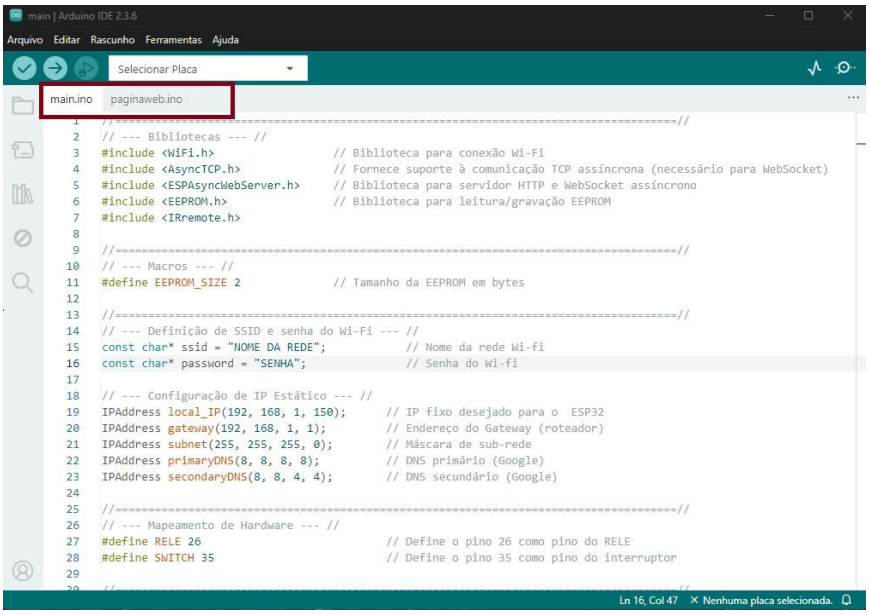
4.4.1 Aspectos Comuns aos Protótipos Desenvolvidos

Nos projetos desenvolvidos, optou-se por organizar o código em dois arquivos distintos:

- ***main.h*:** responsável pela implementação do programa principal.
- ***paginaweb.h*:**: destinado à estrutura da interface web, contendo o código em HTML, CSS, JavaScript e as rotinas de comunicação via WebSocket.

A Figura 16, na parte em vermelho, ilustra essa forma de divisão do código.

Figura 16 – Divisão do Código



Fonte: Autoria Própria

O uso de bibliotecas é fundamental no desenvolvimento de um código, pois elas fornecem recursos prontos que simplificam a programação, reduzem a complexidade do código e permitem
implementar funcionalidades avançadas. Para o funcionamento dos dois protótipos, foram incorporadas as bibliotecas Nos dois protótipos desenvolvidos empregam-se as bibliotecas WiFi.h, *AsyncTCP.h* e *ESPAsyncWebServer.h*.

• **WiFi.h:** estabelece a conectividade sem fio do ESP32, permite que o ESP32 se conecte a uma rede sem fio, configurando nome, senha e endereço de rede.

• ***AsyncTCP.h***: provê comunicação TCP assíncrona, evitando bloqueios e viabilizando múltiplas conexões de forma eficiente.

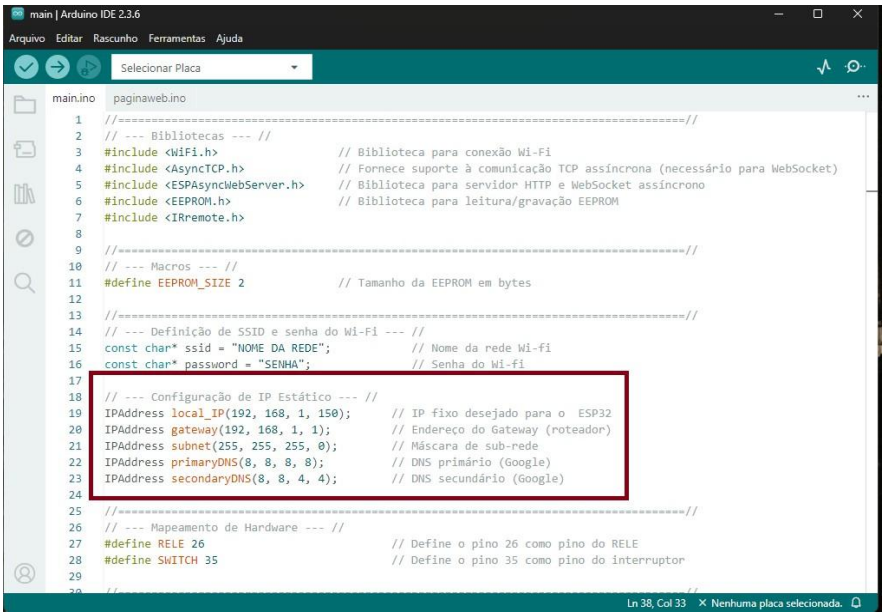
• ***ESPAsyncWebServer.h*****:** possibilita a criação de servidor HTTP e WebSocket não bloqueantes, por meio dos quais ambos os protótipos hospedam a interface acessível via navegador e realizam comunicação em tempo real com o usuário, garantindo maior responsividade e interatividade no controle do sistema.

Na prática, essas três bibliotecas, quando utilizadas em conjunto, permitem que vários usuários se conectem ao servidor simultaneamente. Além disso, qualquer modificação feita por um usuário é imediatamente refletida para

todos os demais, sem a necessidade de recarregar a página.

Outro recurso importante utilizado em ambos os protótipos foi a configuração de IP estático, feita por meio da biblioteca *WiFi.h*. Esse recurso garante que o ESP32 mantenha sempre o mesmo endereço dentro da rede, facilitando o acesso ao servidor web e evitando falhas na comunicação quando ocorrem reconexões. Conforme mostra a Figura 17 na parte em vermelho.

Figura 17 – IP Estático



Fonte: Autoria Própria

4.4.2 Aspector Específicos do Código de Iluminação

No desenvolvimento do código de iluminação, foi necessário adotar um mecanismo que assegurasse a preservação do estado da lâmpada mesmo após falhas de energia ou reinicialização do microcontrolador. Para esse fim, utilizou-se a biblioteca *EEPROM.h*, que permite realizar operações de leitura e escrita em memória não volátil, possibilitando a restauração automática do último estado registrado. No caso do ESP32, essa biblioteca não está associada a um hardware EEPROM dedicado, mas sim a uma emulação realizada na memória flash, implementada sobre a NVS (*Non-Volatile Storage*) do ESP-IDF, o que garante compatibilidade com aplicações Arduino e praticidade no armazenamento de dados persistentes (DEEPBLUEMBEDDED, 2025).

4.4.3 Aspector Específicos do Código de Ar-condicionado

No desenvolvimento do código de iluminação foram utilizadas bibliotecas adicionais que ampliam as funcionalidades do sistema, proporcionando tanto a comunicação via infravermelho quanto o gerenciamento de arquivos em memória não volátil. Cada uma delas exerce um papel específico, conforme descrito a seguir:

• ***IRremote.h*****:** A biblioteca IRremote é responsável pelo envio e recepção de sinais de infravermelho (IR). No contexto do projeto, ela possibilita que o ESP32 capture códigos emitidos por controles remotos convencionais e, posteriormente, os reproduza para acionar dispositivos compatíveis.

• ***FS.h***: A biblioteca FS define uma interface padrão para manipulação de sistemas de arquivos em dispositivos embarcados. No projeto, ela atua como camada de abstração, permitindo criar, ler, escrever e excluir arquivos de forma uniforme, independentemente do sistema de arquivos utilizado.

• ***LittleFS.h***: A biblioteca *LittleFS* implementa o sistema de arquivos LittleFS diretamente na memória flash interna do ESP32. Essa solução substitui o antigo SPIFFS, oferecendo maior confiabilidade na gravação de dados, suporte a diretórios e melhor desempenho em operações frequentes de escrita e leitura.

4.5 Considerações Finais do Capítulo

O capítulo apresentou o desenvolvimento do sistema de automação IoT voltado ao controle de iluminação e ar-condicionado, detalhando tanto os aspectos de conectividade e interface web quanto a organização do código e os recursos utilizados para garantir confiabilidade e usabilidade. Destacou-se a utilização da IDE do Arduino, escolhida por sua simplicidade e ampla documentação, além da integração de bibliotecas que viabilizam comunicação assíncrona, hospedagem da interface e interação em tempo real com múltiplos usuários. Também foram ressaltados mecanismos como a configuração de IP estático e o uso de memória não volátil,

fundamentais para assegurar estabilidade e persistência das informações no funcionamento do sistema.

Nos protótipos desenvolvidos, a implementação de bibliotecas específicas como *EE-PROM.h*, *IRremote.h*, FS.h e *LittleFS.h* demonstrou-se essencial para a ampliação das funcionalidades, permitindo desde o armazenamento de estados da lâmpada até o registro e reprodução de sinais infravermelhos, além da gestão eficiente de arquivos em memória *flash*. Com isso, consolidou-se um sistema robusto, de baixo custo e capaz de integrar dispositivos distintos em uma mesma plataforma de automação, constituindo a base para as análises e resultados apresentados no próximo capítulo.

5 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos a nível de hardware e software com o desenvolvimento deste trabalho.

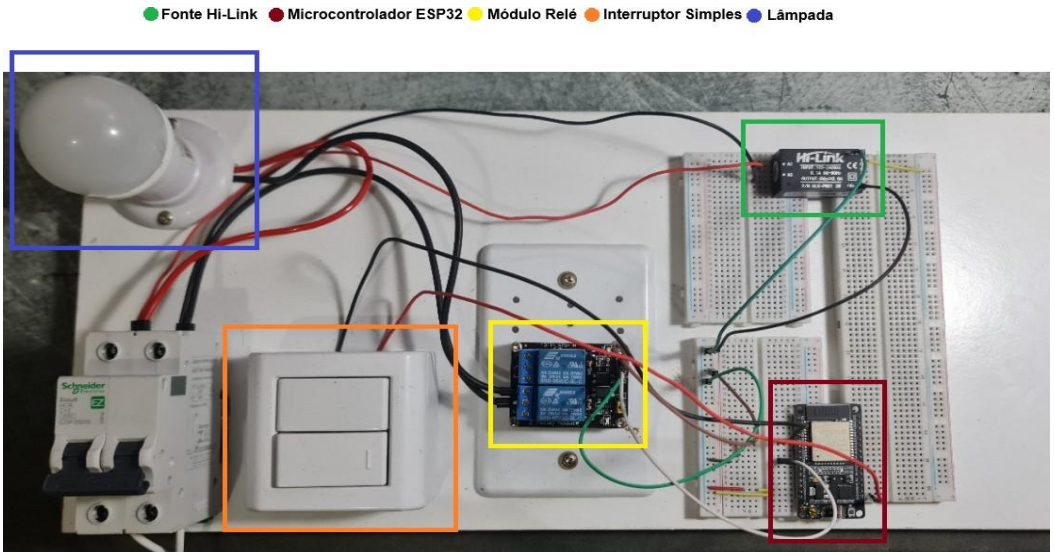
5.1 Sistema de Automação para controle de Iluminação

Nesta seção serão descritos os circuitos de transmissão e recepção por infravermelho, o controle de iluminação via relés e a integração com o ESP32. Também são abordados o *firmware* do microcontrolador, os serviços de comunicação e a interface web, evidenciando como hardware e software foram integrados para viabilizar a operação e validação do sistema.

5.1.1 *Hardware*

O circuito de controle de iluminação foi desenvolvido utilizando um módulo relé eletromecânico de dois canais, um interruptor simples, uma lâmpada de teste e uma fonte *Hi-Link* para a alimentação do sistema, conforme ilustrado na Figura 18. O relé funciona como uma chave eletromecânica intermediária entre o microcontrolador ESP32 e a rede elétrica, permitindo ligar ou desligar a lâmpada de acordo com os comandos enviados pelo sistema.

Figura 18 – Protótipo do Controle de iluminação



Fonte: Autoria Própria

O disjuntor, mostrado na Figura 18, não faz parte do objeto de estudo deste protótipo. Sua presença no protótipo tem finalidade apenas prática, servindo como elemento de proteção e de seccionamento durante a fase de testes. Dessa forma, foi possível ligar e desligar rapidamente o circuito sempre que necessário, garantindo maior segurança e facilidade de manuseio no processo de validação do sistema.

5.1.2 *Software*

O desenvolvimento do software foi realizado na IDE Arduino, utilizando a linguagem C/C++. O código foi estruturado para permitir duas formas de acionamento da lâmpada: pelo interruptor físico e pela interface web. O ESP32 gerencia a lógica de controle, identificando se o sinal de entrada vem do interruptor ou da página web. A interface foi desenvolvida em HTML e disponibilizada através do servidor hospedado no próprio ESP32, possibilitando o controle remoto por dispositivos conectados à mesma rede Wi-Fi. Adicionalmente, foi implementada uma função de armazenamento do último estado da lâmpada na memória EEPROM. Dessa forma, após quedas de energia ou reinicializações, a lâmpada retorna automaticamente ao estado anterior, ampliando a praticidade e a confiabilidade do sistema.

Durante a lógica de desenvolvimento do código, um dos obstáculos enfrentados foi a necessidade de garantir que o acionamento da lâmpada ocorresse de forma confiável, independentemente do nível lógico lido na entrada digital do ESP32. Para superar essa limitação, optou-se por implementar a identificação da mudança de estado do interruptor, em vez de apenas verificar se o sinal estava em nível alto ou baixo. Na prática, a *variável estadoDoSwitchAnt* armazena o nível lógico anteriormente lido pelo ESP32, enquanto a variável leituraSwitch registra o nível lógico atual. Quando esses valores são diferentes, a transição é detectada e o acionamento ocorre.

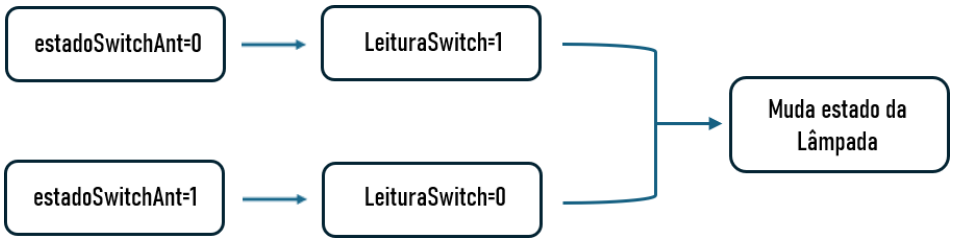
Figura 19 – Mudança de Estado do Interruptor no Código

```
void loop()
{
  leituraSwitch = digitalRead(SWITCH);            // Lê o estado atual do interruptor

  if(estadoSwitchAnt != leituraSwitch)            // Detecta qualquer mudança de estado no interruptor
  {
    if (millis() - ultimaTroca > debounceDelay)
    {
    ultimaTroca = millis();                       // Atualiza o tempo da última mudança

    estadoLED = !estadoLED;                       // Inverte estado do LED
    digitalWrite(RELE, !estadoLED);               // Atualiza o relé
    Serial.println(estadoLED ? "LED ligado via Interruptor" : "LED desligado via Interruptor");  // Imprime mensagem

    salvarEEPROM();

    notifyClients();                              //  Envia o novo estado para a página HTML
    }

  estadoSwitchAnt = leituraSwitch;                // Atualiza o estado anterior do interruptor

  }
}
```

Fonte: Autoria Própria

Na Figura 20 é possível visualizar a mudança de estado em forma de diagrama de fluxo.

Figura 20 – Fluxograma de Mudança de Estado



Fonte: Autoria Própria

Essa análise por mudança de estado mostrou-se uma alternativa elegante para detectar a transição do interruptor. Entretanto, essa abordagem trouxe um problema: todas as variáveis de estado são iniciadas com valor igual a zero no início da execução do código. Esse comportamento inicial está ilustrado na Figura 21, que apresenta o trecho do código responsável pela declaração e inicialização dessas variáveis.

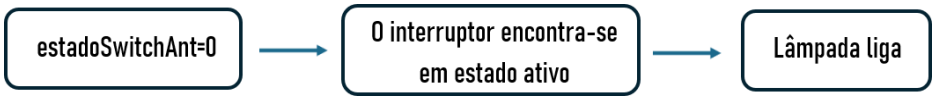
Figura 21 – Variáveis de Estado do Código

```
//=============================================================================//
// --- Variáveis de Estado --- //
int estadoSwitchAnt = 0;  // Armazena o estado anterior do interruptor
int permiteSalvarAnt=0;   // Armazena o estado anterior da seleção de armazenamento da memória flash
int leituraSwitch = 0;    // Armazena a leitura atual do interruptor
int permiteSalvar=0;      // Armazena a leitura atual da seleção de armazenamento da memória flash
int estadoLED = 0;        // Estado atual do LED
```

Fonte: Autoria Própria

No momento da inicialização, caso o interruptor se encontre na posição comutada, o ESP32 detectará nível lógico alto em sua entrada digital. Como consequência, o sistema interpretará esse estado como um acionamento válido e a lâmpada será ligada automaticamente. Como mostrado no diagrama de fluxo da Figura 22.

Figura 22 – Fluxograma do Interruptor Comutado



Fonte: Autoria Própria

A solução encontrada, apresentada na Figura 23, consistiu em corrigir o estado inicial do interruptor no momento da energização do sistema. Para isso, o código foi ajustado de modo a realizar uma leitura imediata da porta digital e, caso fosse detectado nível lógico alto, atualizar a variável de estado (*estadoSwitchAnt*). Esse procedimento garante que o valor armazenado represente fielmente a condição real do interruptor, evitando que a lâmpada seja acionada de forma incorreta logo na inicialização.

Figura 23 – Correção do Estado Inicial do Interruptor

```
// ------------------------------------------------------------- //
// --- Correção do estado inicial do interruptor, se necessário --- //
leituraSwitch = digitalRead(SWITCH);
if(estadoSwitchAnt == 0 && leituraSwitch == 1)
{
  estadoSwitchAnt = 1;
}
```

Fonte: Autoria Própria

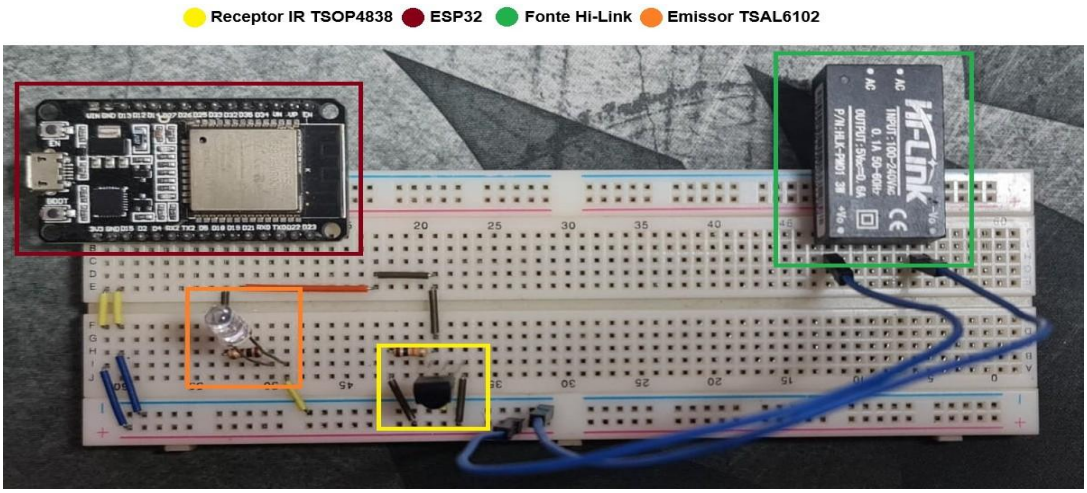
5.2 Sistema de Automação para controle de Ar-Condicionado

5.2.1 *Hardware*

O protótipo de controle do ar-condicionado foi desenvolvido a partir da integração de quatro elementos principais de hardware. O microcontrolador ESP32 atua como unidade central de processamento, responsável por executar a lógica do sistema e gerenciar a comunicação com a interface web, conforme mostrado na Figura. A transmissão dos comandos para o ar-condicionado é realizada por meio do LED emissor infravermelho TSAL6102, enquanto a recepção dos sinais é garantida pelo sensor TSOP4838, capaz de identificar

códigos de controle remoto na faixa de 38 kHz. Para a alimentação do conjunto, utilizou-se a fonte Hi-Link, que converte a tensão da rede elétrica em 5 V, fornecendo energia estável e segura para todo o circuito. Conforme mostra a Figura 24.

Figura 24 – Protótipo do Controle do ar-condiconado



Fonte: Autoria Própria

5.2.2 *Software*

O software do protótipo do ar-condicionado foi implementado em C/C++ na IDE Arduino, fazendo uso das bibliotecas *IRremote*, *WiFi.h*, *AsyncTCP.h* e *ESPAsyncWebServer.h*. O código foi projetado para desempenhar duas funções principais: envio e registro de sinais infra vermelhos. No envio, os sinais previamente armazenados em vetores (*rawData*) são transmitidos ao equipamento, permitindo funções como ligar/desligar e alteração de temperatura. No registro, o sistema possibilita capturar novos sinais a partir do controle remoto original, armazenando-os para uso posterior.

Um dos obstáculos superados durante o desenvolvimento foi a necessidade de lidar com o tamanho extenso dos vetores rawData, que representam os sinais infravermelhos do controle remoto. Inicialmente, havia a limitação de salvá-los diretamente na memória EEPROM, devido à sua baixa capacidade. Para contornar esse problema, adotou-se o uso da biblioteca <*LittleFS.h*>, que possibilita gravar e gerenciar arquivos dentro da memória flash do ESP32. Essa solução tornou viável o armazenamento e a posterior leitura

dos códigos de sinais, garantindo a confiabilidade do processo de registro e reuso.

5.3 Considerações Finais do Capítulo

Neste capítulo foram apresentados os resultados dos sistemas de automação desenvolvidos. No controle de iluminação, o ESP32, associado a relés e a uma interface web, permitiu o acionamento remoto e físico da lâmpada, com armazenamento do último estado na EEPROM e correção de falhas na inicialização por meio da detecção de mudança de estado do interruptor. Já no controle do ar-condicionado, a integração do ESP32 com emissor e receptor infravermelho possibilitou o envio e registro de sinais do controle remoto, superando a limitação da EEPROM com o uso da biblioteca <*LittleFS.h*> para armazenar vetores extensos. Os resultados demonstram a viabilidade técnica e a confiabilidade dos protótipos, reforçando o potencial de aplicação em soluções de automação residenciais e comerciais.

6 CONCLUSÃO

6.1 Limitações e Desafios

O desenvolvimento dos protótipos evidenciou alguns desafios técnicos e limitações práticas. No sistema de iluminação, destacou-se a necessidade de corrigir o estado inicial do interruptor, evitando acionamentos indevidos, além da atenção a possíveis interferências e à necessidade de debouncing. Já no sistema de ar-condicionado, a principal dificuldade foi o armazenamento dos sinais infravermelhos em função do tamanho dos vetores *rawData*, solucionada com o uso da biblioteca <*LittleFS.h*>. Uma limitação importante observada em ambos os protótipos é a dependência da rede Wi-Fi: em ambientes com instabilidade ou queda de conexão, o controle remoto por interface web torna-se indisponível, restringindo o acionamento ao modo físico. Essa característica reduz a confiabilidade em contextos onde a rede não possui boa cobertura, configurando-se como ponto a ser aprimorado em trabalhos futuros.

6.1.1 Trabalhos Futuros

Como continuidade deste trabalho, algumas melhorias podem ser exploradas para ampliar a robustez e a escalabilidade dos sistemas desenvolvidos:

- Integração com o servidor externo: hospedar a interface de interação com o usuário diretamente no servidor, centralizando o gerenciamento e reduzindo a dependência da memória interna do ESP32.

- Comunicação via MQTT: implementar a comunicação com um servidor externo utilizando o protocolo MQTT, permitindo maior interoperabilidade, escalabilidade e integração com outras soluções de automação residencial ou comercial.

6.2 Considerações Finais do Capítulo

O presente Trabalho de Conclusão de Curso teve como objetivo o desenvolvimento de um sistema de automação baseado em Internet das Coisas

(IoT) para o controle de iluminação e de ar-condicionado. Utilizando o microcontrolador ESP32 como unidade central, foi possível integrar hardware e software em protótipos funcionais, validados em ambiente de teste.

No sistema de iluminação, a utilização de relés permitiu o acionamento da lâmpada tanto por interruptor físico quanto por interface web, com a adição do recurso de armazenamento do último estado em memória EEPROM, garantindo praticidade e confiabilidade. No sistema de ar-condicionado, a transmissão e o registro de sinais infravermelhos possibilitaram simular o funcionamento de um controle remoto convencional, superando a limitação da memória EEPROM por meio da adoção da biblioteca <*LittleFS.h*>.

Os resultados obtidos confirmam a viabilidade técnica e o baixo custo da solução proposta, demonstrando que sistemas de automação simples podem ser implementados com eficiência em ambientes residenciais ou comerciais. Entretanto, foram identificadas limitações, como a dependência da rede Wi-Fi e a necessidade de maior robustez no tratamento de sinais de entrada, que abrem espaço para futuros aprimoramentos.

Por fim, este trabalho contribui para a democratização da automação, oferecendo uma alternativa prática, acessível e expansível, ao mesmo tempo em que estabelece uma base sólida para pesquisas futuras, como a integração com o servidor externo e o uso de comunicação MQTT.

6.3 Declaração do uso de Inteligência Artificial

Durante o desenvolvimento deste trabalho, foi utilizada a Inteligência Artificial (IA) para auxiliar em algumas etapas da pesquisa e elaboração, conforme segue:

1. **Ferramenta de IA utilizada:** Chat GPT

2. **Objetivo do uso:** A IA foi empregada para auxiliar na elaboração de resumos, sugestões de tópicos, revisão gramatical, organização de referências bibliográficas e aprimoramento da clareza textual.

3. **Processo de integração da IA:** A ferramenta foi utilizada de forma pontual, com o propósito de gerar sugestões linguísticas, estruturar trechos do

texto e verificar a consistência de informações.

A autoria do conteúdo original e as análises realizadas neste trabalho são de responsa bilidade exclusiva do autor. A utilização da IA não substitui a análise crítica, interpretação ou conclusões da pesquisa. A IA foi empregada como ferramenta auxiliar para facilitar algumas etapas do processo de escrita e análise. Declaro, portanto, que o uso da IA foi realizado dentro das normas éticas e acadêmicas, com o intuito de melhorar a qualidade e precisão do trabalho, sem comprometer a integridade e a originalidade do conteúdo.

REFERÊNCIAS

ALVES, Adriano Ferreira; COSTA, André Henrique Muniz; SANÇÃO, Gustavo Sousa; LIMA, Lucca Santos de Oliveira. Automação residencial: evolução da segurança e conforto na sua moradia. *Revista Científica Multidisciplinar O Saber*, São Paulo, v. 2, n. 2, p. 45–56, jul./dez. 2022. Disponível em: https://submissoesrevistarcmos.com.br/index.php/rcmos/article/download/310/296. Acesso em: 21 set. 2025.

AMÉRICO, Jonas de Jesus. **Automação residencial de baixo custo.** Criciúma: UNESC, [s.d.]. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade do Extremo Sul Catarinense. Disponível em: https://repositorio.unesc.net/bitstream/1/8860/1/Jonas%20de%20Jesus%20Am%C3%A9rico.pdf. Acesso em: 29 set. 2025.

ASHTON, Kevin. That 'Internet of Things' Thing. *RFID Journal*, 2009. Disponível em: https://www.rfidjournal.com/articles/view?4986. Acesso em: 19 set. 2025.

BANKS, Andy; GUPTA, Rahul. **MQTT Version 3.1.1.** *OASIS Standard*, 2014. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Acesso em: 19 set. 2025.

CHANCUCO. **PROTEUS-LIBRARY-ESP32-DEVKIT.** [Repositório GitHub]. Disponível em: https://github.com/CHANCUCO/PROTEUS-LIBRARY-ESP32-DEVKIT. Acesso em: 23 set. 2025.

COELHO, D. F. B.; CRUZ, V. H. D. N. **Edifícios inteligentes: uma visão das tecnologias aplicadas.** São Paulo: Editora Blucher, 2017.

DOS SANTOS, Jhennifer F. et al. SAIR: Sistema Automatizado de Iluminação e Refrigeração. *Simpósio Brasileiro de Sistemas Elétricos – SBSE*, v. 2, n. 1, 2022.

ESPRESSIF. **ESP-WROOM-32 Datasheet.** Versão 2.9. 2019.

ESPRESSIF. **Esp32 Overview.** Espressif, 2019. Disponível em: https://www.espressif.com/en/products/hardware/esp32/overview. Acesso em: mar. 2025.

ESPRESSIF SYSTEMS. **ESP32 Series Datasheet.** Shanghai: Espressif Systems, 2016. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 9 set. 2025.

ESPRESSIF SYSTEMS. **ESP32 Series of SoCs.** Shanghai: Espressif Systems, 2021. Disponível em: https://www.espressif.com/en/products/socs. Acesso em: 19 set. 2025.

FIELDING, Roy T.; RESCHKE, Julian F. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.** *IETF RFC 7230*, 2014. Disponível em: https://www.rfc-editor.org/rfc/rfc7230. Acesso em: 19 set. 2025.

FETTE, Ian; MELNIK, Alexey. **The WebSocket Protocol.** *IETF RFC 6455*, 2011. Disponível em: https://www.rfc-editor.org/rfc/rfc6455. Acesso em: 19 set. 2025.

FLANAGAN, David. **JavaScript: o guia definitivo.** 6. ed. Tradução de João Eduardo Nóbrega Tortello. Porto Alegre: Bookman, 2013.

HLKTECH. **Hi-Link Power Modules.** [S.l.], 2023. Disponível em: https://hlktech.net/index.php?id=105. Acesso em: 19 set. 2025.

HUNKELER, Urs; TRUONG, Hong Linh; STANFORD-CLARK, Andy. MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In: *Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE 2008).* Bangalore: IEEE, 2008. p. 791–798.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO); INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC). **ISO/IEC 30141: Internet of Things (IoT) — Reference Architecture.** Geneva: ISO/IEC, 2018.

INTERNATIONAL TELECOMMUNICATION UNION (ITU). **The Internet of Things.** Geneva: ITU, 2005. Disponível em: https://www.itu.int/osg/spu/publications/internetofthings/. Acesso em: 19 set. 2025.

JR, S. L. S.; FARINELLI, F. A. **Domótica – Automação residencial e casas inteligentes com Arduino e ESP8626.** São Paulo: Editora Saraiva, 2018.

LOBATO, Elen Priscila de Souza. **Desenvolvimento de um sistema IoT para o controle de iluminação residencial baseado nos princípios da Indústria 4.0.** 2019. Trabalho de Conclusão de Curso (Engenharia da Computação e Telecomunicação) – Universidade Federal do Pará, Belém, 2019.

MARTINS, Fábio Augusto; RIBEIRO, Mariana Lopes. Casas inteligentes e Internet das Coisas: tendências e aplicações. *Revista Brasileira de Inovação Tecnológica*, São Paulo, v. 10, n. 1, p. 34–49, 2021.

MERÇON, Victor de Abreu. **Sistema de controle de iluminação e monitoramento de consumo elétrico residencial via aplicativo.** 2022. 100 f. Monografia (Graduação em Engenharia de Controle e Automação) – Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2022.

MERCADO LIVRE. **Módulo relé 2 canais 5V Arduino PIC Raspberry Pi.** [S.l.], 2025. Disponível em: https://www.mercadolivre.com.br/modulo-rele-rele-

2-canais-5v-arduino-pic-raspberry-pi/p/MLB32973442. Acesso em: 19 set. 2025.

MOHANAN, Vishnu. **DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram and Reference.** *Circuitstate Electronics*, 20 dez. 2022. Disponível em: https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/. Acesso em: 28 set. 2025.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). **Recommendations for IoT device manufacturers: NISTIR 8259.** Gaithersburg, MD: NIST, 2020. Disponível em: https://csrc.nist.gov/publications/detail/nistir/8259/final. Acesso em: 19 set. 2025.

OASIS. **MQTT Version 5.0.** 2019. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Acesso em: 19 set. 2025.

OLIVEIRA, L.; MORAIS, A. Desenvolvimento de soluções em IoT utilizando ESP32. *Revista de Sistemas e Computação*, v. 10, n. 2, p. 45–59, 2020.

OLIVEIRA. **ESP32 e suas versões.** [S.l.], 2019. Disponível em: https://xprojetos.net/esp32-e-suas-versoes/. Acesso em: 19 set. 2025.

REVISTA FT. Estudo de caso de automação residencial baseada em Internet das Coisas. *Revista FT*, v. 5, n. 6, p. 1717–1726, 2023. Disponível em: https://revistaft.com.br/estudo-de-caso-de-automacao-residencial-baseada-em-internet-das-coisas/. Acesso em: 29 set. 2025.

SHELBY, Zach et al. **The Constrained Application Protocol (CoAP).** *IETF RFC 7252*, 2014. Disponível em: https://www.rfc-editor.org/rfc/rfc7252. Acesso em: 19 set. 2025.

SILVA, Bruno Henrique; PEREIRA, Carlos Eduardo. Aplicações de baixo custo em automação residencial com ESP32. *Revista de Engenharia e Pesquisa Aplicada*, Belo Horizonte, v. 5, n. 2, p. 77–85, 2020.

SILVA, Jorge Fernando de Barros. **Desenvolvimento de sistema de segurança residencial com ESP32 e monitoramento via Telegram.** Recife, 2023. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal de Pernambuco. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/56565/4/Monogra. Acesso em: 29 set. 2025.

SOARES, Fernando Gomes. **Desenvolvimento de um protótipo demonstrativo de automação residencial, utilizando microcontrolador ESP32.** Campo Grande, 2023. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal de Mato Grosso do Sul. Disponível em: https://engeletrica-faeng.ufms.br/files/2023/08/TCC-LGJ-AutomacaoResidencial-FernandoGomes-2023.pdf. Acesso em: 29 set. 2025.

SOUZA, Jorge Luiz de; OLIVEIRA, João Carlos. **Internet das Coisas: fundamentos e aplicações.** São Paulo: Saraiva Educação, 2019.

VISHAY INTERTECHNOLOGY. **TSOP48 Series – Miniature IR Receiver Modules for Remote Control Systems.** Datasheet. Malvern, 2017. Disponível em: https://www.vishay.com/docs/82459/tsop48.pdf. Acesso em: 19 set. 2025.

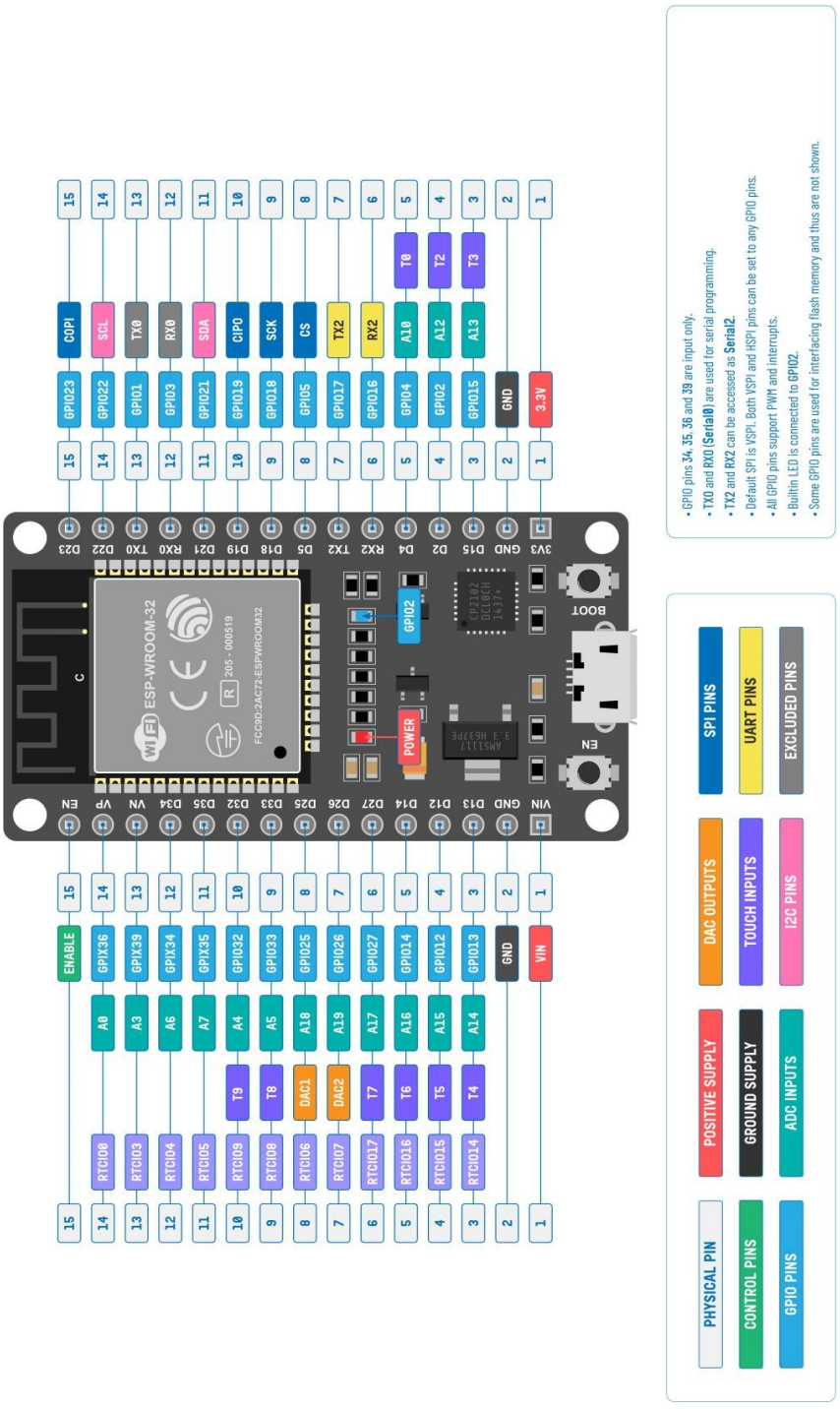
WHITE, Neil Kolban. **Kolban's Book on ESP32.** [S.l.]: Leanpub, 2017. Disponível em: https://leanpub.com/kolban-ESP32. Acesso em: 29 set. 2025.

WORLD WIDE WEB CONSORTIUM (W3C). **Cascading Style Sheets Level 3 (CSS3) Specification.** *W3C Recommendation*, 2015. Disponível em: https://www.w3.org/TR/css-2015/. Acesso em: 19 set. 2025.

WORLD WIDE WEB CONSORTIUM (W3C). **HTML5: A vocabulary and associated APIs for HTML and XHTML.** *W3C Recommendation*, 2017. Disponível em: https://www.w3.org/TR/html5/. Acesso em: 19 set. 2025.

ZANATTA, Bianca. Casa "inteligente" é cada vez mais realidade: Associação avalia que uso de dispositivos de automação nas residências deve crescer 20% até 2023. *O Estado de São Paulo*, jan. 2021. Disponível em: https://www.estadao.com.br/economia/radar-imobiliario/casa-inteligente-e-cada-vez-mais-realidade/. Acesso em: 9 set. 2025.

ANEXO A — Pinout Módulo ESP32 DEV KIT V1



Fonte: (MOHANAN, 2022) – Adaptado.