UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

FRANCISCO CAIOÃ DE ARAGÃO RODRIGUES

ASSISTENTE VIRTUAL EMBARCADA

SOBRAL

2026

FRANCISCO CAIOÃ DE ARAGÃO RODRIGUES

ASSISTENTE VIRTUAL EMBARCADA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Wendley Souza da Silva

SOBRAL

2026

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R613a Rodrigues, Francisco Caioã de Aragão.
Assistente Virtual Embarcada / Francisco Caioã de Aragão Rodrigues. – 2026.
50 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2026.
Orientação: Prof. Dr. Wendley Souza da Silva.

1. Assistente virtual. 2. Redes neurais artificiais. 3. Processamento de linguagem natural. I. Título.
CDD 621.39

FRANCISCO CAIOÃ DE ARAGÃO RODRIGUES

ASSISTENTE VIRTUAL EMBARCADA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 28/01/2026

BANCA EXAMINADORA

Prof. Dr. Wendley Souza da Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. Danillo Fernandes do Nascimento
Universidade Federal do Ceará (UFC)

Me. Geraldo Eufrazio Martins Júnior
Universidade Federal do Ceará (UFC)

Dedico este trabalho aos meus pais, por todos os ensinamentos, sacrifícios e pelo amor que me trouxe até aqui. Sem vocês, nada disso seria possível.

AGRADECIMENTOS

A Cristo, que pagou pelos meus pecados na cruz do calvário. Seu sacrifício foi único e eficaz. E por conta disso, fui liberto da escravidão do pecado.

Aos meus pais, Francisco Teófilo Rodrigues e Maria Alexsandra Souza de Aragão, que me deram a vida e sempre estiveram comigo.

À minha irmã, Maria Caioanny de Aragão Rodrigues, que tem me encorajado.

À minha avó, Maria Auxiliadora de Sousa e meus tios Teresinha Michelle de Sousa Aragão, Francisca Adriana de Sousa Aragão e Benedito Tiago de Sousa Aragão, que acreditaram em mim durante toda a minha graduação.

Ao professor Dr. Wendley Souza da Silva, que me incentivou a continuar desenvolvendo este projeto e orientou com sabedoria e motivação.

“ Louvem o Senhor, todas as nações; exaltem-no, todos os povos! Porque imenso é o seu amor leal por nós, e a fidelidade do Senhor dura para sempre. Aleluia! ”

(Salmos 117:1,2)

RESUMO

A Assistente Virtual Embarcada (EVA) foi pesquisada, projetada, implementada e validada com o propósito de oferecer uma solução de automação residencial de baixo custo, que prioriza o processamento local de dados e o controle direto de dispositivos domésticos, minimizando a dependência de nuvem e os riscos à privacidade do usuário. A arquitetura do sistema segue um paradigma de computação distribuída em borda, dividindo o processamento entre um módulo de aquisição (baseado no microcontrolador ESP-32-WROOM-32) e um módulo de processamento instalado em uma rede de computadores local. O módulo de aquisição é responsável pela captura de áudio através de microfone e pela reprodução de feedback sonoro, ativando-se apenas ao pressionar um botão e comunicando-se via Wi-Fi. Os dados coletados são enviados para o servidor em borda via protocolo TCP/IP. O módulo de processamento, implementado em Python, executa tarefas computacionalmente intensivas como transcrição de áudio para texto utilizando um framework de redes neurais artificiais (Vosk) e processamento de linguagem natural para identificação de intenções e gerenciamento de dispositivos. Após o processamento, os resultados são retornados ao microcontrolador em formato JSON para execução de ações e feedback ao usuário. A metodologia incluiu estudo bibliográfico sobre Redes Neurais Artificiais para áudio, análise comparativa de projetos similares e validação por testes manuais do protótipo. Espera-se que o sistema interprete comandos de voz simples, como automação de iluminação ou respostas a perguntas, com alta acurácia. Este trabalho demonstra a viabilidade técnica e econômica da EVA, combinando eficiência energética com capacidade de processamento avançado. A arquitetura modular e o foco no processamento local otimizam recursos, atendem às demandas por privacidade e autonomia, e facilitam expansão e customização para diversos cenários de aplicação, com protocolos de segurança que garantem operação confiável.

Palavras-chave: Assistente virtual; Embarcado; Automação residencial; Edge computing; Privacidade; ESP-32; Redes neurais artificiais; Processamento de linguagem natural.

ABSTRACT

The Embedded Virtual Assistant (EVA) was researched, designed, implemented, and validated with the aim of offering a low-cost home automation solution that prioritizes local data processing and direct control of domestic devices, minimizing cloud dependency and user privacy risks. The system's architecture follows an edge computing paradigm, dividing processing between an acquisition module (based on the ESP-32-WROOM-32 microcontroller) and a processing module installed on a local computer network. The acquisition module is responsible for audio capture via microphone and sound feedback reproduction, activating only when a button is pressed and communicating via Wi-Fi. Collected data is sent to the edge server via TCP/IP. The processing module, implemented in Python, performs computationally intensive tasks such as audio-to-text transcription using an artificial neural network framework (Vosk) and natural language processing for intention identification and device management. After processing, results are returned to the microcontroller in JSON format for action execution and user feedback. The methodology included a bibliographic study of Artificial Neural Networks for audio, comparative analysis of similar projects, and validation through manual prototype testing. The system is expected to interpret simple voice commands, such as lighting automation or answering questions, with high accuracy. This work demonstrates EVA's technical and economic feasibility, combining energy efficiency with advanced processing capabilities. The modular architecture and focus on local processing optimize resources, meet demands for privacy and autonomy, and facilitate expansion and customization for various application scenarios, with security protocols ensuring reliable operation.

Keywords: Virtual assistant; Embedded; Home automation; Edge computing; Privacy; ESP-32; Artificial neural networks; Natural language processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

AV	Assistente Virtual
AVE	Assistente Virtual Embarcada
IVA	*Intelligent Virtual Assistant*
JSON	*JavaScript Object Notation*
RTOS	*Real Time Operating System*
MCU	*Microcontroller Unit*
CPU	*Central Processing Unit*
PLN	Processamento de Linguagem Natural
1²S	*Inter-Integrated Circuit Sound*
NLU	*Natural Language Understanding*
HRI	*Human-Robot Interaction*
TIC	Tecnologias de Informação e Comunicação
TCP/IP	*Transmission Control Protocol/Internet Protocol*

SUMÁRIO

1 INTRODUÇÃO

Assistentes virtuais (AVs) têm se tornado cada vez mais presentes no cotidiano, transformando a forma as pessoas interagem com dispositivos eletrônicos, sistemas residenciais e aplicações de voz Matos e Oliveira (2021). Soluções comerciais amplamente utilizadas, como *Amazon Alexa*, *Google Assistant* e *Apple Siri*, exemplificam o potencial dessa tecnologia ao oferecer conveniência e eficiência, mas também apresentam importantes limitações, especialmente no que tange à privacidade dos dados, dependência de conectividade com a nuvem e custos elevados para implementação em larga escala (MATOS; OLIVEIRA, 2021). Essas plataformas armazenam gravações de voz e informações de atividade na nuvem, abrindo brechas de segurança e gerando preocupações quanto ao controle e uso dos dados dos usuários (CHUNG; LEE, 2018).

A atuação das AVs ganhou especial relevância no contexto da Quarta Revolução Industrial (Indústria 4.0). As Tecnologias de Informação e Comunicação (TIC) estabelecem-se como propulsoras da melhoria dos processos industriais, fornecendo sistemas digitais que oferecem informações em tempo real e aumentam a flexibilidade e a eficiência. A assistência técnica, um dos princípios de design da indústria 4.0, define o papel vital dos assistentes virtuais no processamento de dados de produção e na oferta de informações contextuais aos trabalhadores (PEREIRA *et al.*, 2023).

Uma AV pode ser conceituada como uma agente de software capaz de executar tarefas ou fornecer serviços em resposta a comandos ou perguntas do usuário (PATIL *et al.*, 2023). É crucial, todavia, estabelecer a distinção entre uma AV e um *chatbot*. Os *chatbots* são programas desenvolvidos para interagir com humanos via texto ou voz, representando uma interface gráfica. As AVs, por sua vez, são soluções de *software* mais amplas, capazes de executar tarefas complexas de forma autônoma com base na entrada de dados realizada por humanos, podendo fazer o uso ou não das interfaces de *chatbot* (PEREIRA *et al.*, 2023).

Nesse contexto, emerge a proposta de desenvolvimento da EVA (do inglês, *Embedded Virtual Assistant* ou em português, Assistente Virtual Embarcada), um sistema de automação residencial de arquitetura distribuída, que busca aliar eficiência energética a microcontroladores com a capacidade de computação em borda (do inglês, *edge computing*) presente em uma rede de computadores local.

A proposta concentra-se na criação de uma plataforma que ofereça baixo custo, proteção à privacidade e eficiência energética, aproveitando os recursos do *ESP-32* e arquiteturas distribuídas. O sistema pretende ainda oferecer respostas rápidas a comandos pré definidos pelo

projetista e controle direto de dispositivos acoplados ao módulo de processamento, reforçando a autonomia do usuário em seu ambiente.

Este trabalho, portanto, contribui para a literatura de assistentes virtuais embarcados ao apresentar uma solução híbrida, alinhada à crescente demanda por sistemas que sejam ao mesmo tempo eficientes e ajudem a manter a privacidade dos usuários (PEREIRA *et al.*, 2023). A arquitetura distribuída e a utilização de microcontroladores como o ESP-32 possibilitam operar com menor latência e maior controle sobre os dados. Isso minimiza a dependência da infraestrutura de nuvem e mitiga as limitações de soluções comerciais tradicionais, como a ausência de exportação de registros de uso para monitoramento do sistema e usuário.

1.1 *Internet of Things* - IoT

A Internet das Coisas (IoT) é um conceito que está evoluindo rapidamente e demonstrando um crescimento significativo no panorama tecnológico e econômico global. Analistas projetam que o mercado mundial de IoT alcançará a marca de US$ 1,7 trilhão em 2020, impulsionado por uma taxa de crescimento anual composta (CAGR) de 16,9% (CHUNG; LEE, 2018). Nesse cenário, assistentes virtuais baseados em comandos de voz possuem um grande potencial para se tornar amplamente utilizados em diversos dispositivos conectados, operando majoritariamente sobre arquiteturas de computação (CHUNG; LEE, 2018) em nuvem para garantir desempenho e gerenciamento eficiente dos dados.

1.2 A Indústria 4.0 e a Internet das Coisas (IoT)

A Indústria 4.0 define o atual cenário de inovação tecnológica, caracterizando a Quarta Revolução Industrial por meio da automatização e da intensa troca de dados, transformando ambientes industriais e sociais em redes de conexões inteligentes (RIBEIRO, 2023; DIAS; FILHO, 2018). Esse avanço foi impulsionado pela integração da Internet das Coisas (IoT) e da *Internet of Services* (IoS) nos processos de manufatura (DIAS; FILHO, 2018). Nesse contexto, a IoT destaca-se como a infraestrutura essencial de comunicação, possibilitando a conectividade entre objetos físicos e ambientes virtuais, inclusive em aplicações domésticas e urbanas com o uso de assistentes virtuais para o controle de dispositivos inteligentes (DIAS; FILHO, 2018; RIBEIRO, 2023).

1.3 Objetivos Gerais

Este trabalho tem como propósito pesquisar, projetar, implementar e validar uma assistente virtual embarcada — denominada EVA — baseada em um módulo ESP-32-WROOM-32, capaz de capturar comandos de voz por meio de um microfone, enviá-los para um servidor local que simula computação de borda, processá-los (por meio de algoritmos que implementam as funcionalidades *speech-to-text* e compreensão semântica) e retornar uma resposta em formato JSON para o ESP-32 executar ações por meio de um alto-falante ou periféricos conectados.

1.4 Objetivos Específicos

1. Selecionar e integrar os componentes (ESP-32, microfone digital, amplificador de áudio e alto-falante) em um circuito embarcado compacto.
2. Implementar no ESP-32 rotinas de captura de áudio e formatação dos dados para transmissão ao servidor.
3. Criar um serviço que seja executado em uma rede local de computadores que receba sinais de áudio, realize reconhecimento de fala (*speech-to-text*) e interpretação semântica das intenções do usuário.
4. Aplicar cenários de uso reais (ex.: “ligar luz”, “tocar música”, “qual é a temperatura?”) para verificar a precisão, confiabilidade e experiência do usuário.
5. Elaborar manual de uso, diagramas de arquitetura e sugerir melhorias futuras para aplicação em cenários de IoT e automação residencial.

2 FUNDAMENTAÇÃO TEÓRICA

Será apresentada nesta sessão a fundamentação teórica necessária para o entendimento e desenvolvimento deste projeto.

2.1 O Aprendizado de Máquina

Para dar sentido e gerar ações a partir da vasta quantidade de dados transmitidos pela IoT, é essencial o uso do aprendizado de máquina (do inglês, *Machine Learning* (ML)). O ML, um campo da Inteligência Artificial (IA), corresponde a um conjunto de técnicas que permite às máquinas aprenderem a partir dos dados e tomarem decisões de maneira autônoma e sem serem explicitamente programadas para cada ação (RIBEIRO, 2023). Nos AVs, o ML é central para o Processamento de Linguagem Natural (PLN), permitindo que o assistente compreenda comandos de voz com sotaques, gírias e diferentes entonações, além de analisar o histórico de interações e preferências para oferecer recomendações personalizadas (RIBEIRO, 2023).

2.2 O Protocolo TCP/IP

Por fim, toda essa troca de dados e interconexão, da Indústria 4.0 ao *AV*, é viável graças ao Protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*). ESCOLA SUPERIOR DE REDES (2020), FORTINET (2025) definem o TCP/IP como o conjunto de protocolos fundamental que rege a comunicação na *Internet*. Ele é responsável por quebrar a informação em pequenos pacotes, assegurar que esses pacotes sejam endereçados (graças ao *IP*) e entregues de forma confiável e na ordem correta (graças ao *TCP*) ao seu destino final. Dessa forma, o *TCP/IP* é o alicerce de rede que garante a comunicação fluida entre o assistente virtual (servidor) e o dispositivo do usuário (cliente).

2.3 O *FreeRTOS*

O *FreeRTOS* é um RTOS livre e de código aberto. Ele serve como um *kernel* e escalonador em tempo real que foi projetado especificamente para ser pequeno o suficiente para rodar em uma MCU para aplicações embarcadas (HE; HUANG, 2014).

Seu propósito central é fornecer a funcionalidade de escalonamento em tempo real, comunicação entre tarefas, análise de tempo e primitivas de sincronização. Ele suporta um

grande número de arquiteturas de microcontroladores subjacentes e se estabeleceu como a plataforma líder de computação em tempo real para microcontroladores. O *FreeRTOS* também é caracterizado como um RTOS com muitos recursos (como criação e gerenciamento de filas, pilhas, troca de mensagens entre tarefas e gerenciamento de memória à partir de bibliotecas já importadas pelo próprio *kernel*), econômico em relação ao consumo de recursos de *hardware* e é bem suportado para o ensino e desenvolvimento de aplicações em tempo real (HE; HUANG, 2014).

Ao contrário do *kernel* original, conhecido como *Vanilla FreeRTOS*, a implementação fornecida pela Espressif para o microcontrolador ESP32 introduz o Multiprocessamento Simétrico (SMP). Esta arquitetura permite que o sistema operacional gerencie simultaneamente os dois núcleos de processamento do dispositivo, denominados *Protocol CPU* (PRO_CPU) e *Application CPU* (APP_CPU).

A arquitetura do ESP32 permite que o escalonador do FreeRTOS gerencie tarefas de forma dinâmica entre os dois núcleos, otimizando o processamento paralelo (ESPRESSIF SYSTEMS, 2024).

O comportamento do escalonador no cenário SMP possibilita a execução de tarefas com ou sem afinidade de núcleo. Enquanto tarefas fixadas através da função xTaskCreatePinnedToCore() garantem o determinismo para processos críticos, tarefas criadas sem afinidade permitem que o *kernel* distribua a carga de processamento dinamicamente, otimizando o *throughput* global do sistema (ESPRESSIF SYSTEMS, 2024). Além disso, a gestão de recursos de memória no ambiente ESP-IDF é distinta, visto que o tamanho do *stack* das tarefas é definido estritamente em bytes, exigindo um controle rigoroso para evitar o transbordamento de pilha (*stack overflow*) em aplicações de tempo real.

2.4 A Linguagem *Python*

Python é uma linguagem de programação de alto nível comumente usada para processamento de dados e projetos que englobam algoritmos de redes neurais artificiais. Neste projeto, a mesma será utilizada para receber e enviar requisições do módulo microcontrolador por meio de *websockets*, processar os dados recebidos em uma rede neural que é executada em um ambiente local e retornar o resultado do processamento em um tempo quase que instantâneo para o usuário final. A escolha dessa linguagem de programação para o *backend* para este projeto é justificada por seu robusto ecossistema de bibliotecas para ML e processamento de linguagem

natural, como revisado por (PATIL *et al.*, 2023), facilitando a implementação de *frameworks* como o *Vosk* para transcrição de áudio em tempo real.

2.5 Computação em Borda e Computação Distribuída

A crescente integração da *Internet of Things* (*IoT*) e dos Assistentes Virtuais *(AVs)* demanda modelos de processamento que superem as limitações da arquitetura tradicional baseada na Nuvem (*Cloud Computing*). Nesse cenário, a computação em borda (*Edge Computing*) e a Computação Distribuída emergem como soluções cruciais para a otimização de sistemas inteligentes.

A Computação Distribuída é um paradigma onde múltiplos componentes de um sistema, localizados em diferentes computadores em rede, trabalham em conjunto, comunicando-se e coordenando suas ações por meio de troca de mensagens, a fim de realizar uma tarefa comum (COUTINHO, 2021). Este modelo se opõe ao processamento centralizado, visando a melhoria do desempenho, a tolerância a falhas e a eficiência na alocação de recursos.

A computação em borda é uma forma especializada de computação distribuída. Ela representa um modelo que move a capacidade de processamento de dados e os serviços para a "borda"da rede, ou seja, para o mais perto possível da fonte geradora de dados, como os sensores e dispositivos *IoT* (BENDEL, 2018). Enquanto a Nuvem centraliza o processamento em grandes *data centers* distantes, a computação em borda o distribui, permitindo que a análise de dados e a tomada de decisão ocorram localmente.

Essa arquitetura descentralizada é vital para aplicações de Assistentes Virtuais e sistemas de *IoT* que exigem latência mínima, como o controle de equipamentos em tempo real na Indústria 4.0 (RIBEIRO, 2023) ou a resposta imediata de um *AV* a um comando de voz. (GOMES, 2023) detalha que a computação em borda reduz a carga sobre a rede de comunicação, pois evita que grandes volumes de dados brutos sejam enviados para a Nuvem, processando apenas informações críticas localmente e enviando à Nuvem apenas o essencial ou os resultados consolidados. Isso melhora não apenas a velocidade de resposta, mas também a segurança e a privacidade, ao manter dados sensíveis na fonte.

- computação em borda (*Edge Computing*): foca na proximidade física do processamento em relação ao dispositivo (*endpoint*) (BENDEL, 2018).
- Computação Distribuída: foca na cooperação lógica entre múltiplos nós de processamento em uma rede para completar uma tarefa (COUTINHO, 2021).

2.6 Componentes Utilizados

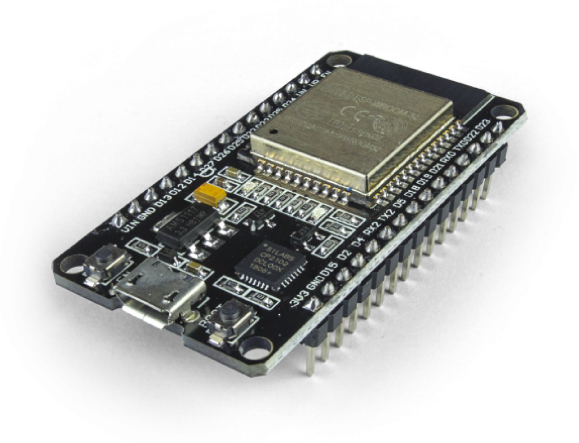
Nesta sessão serão apresentados os componentes de hardware utilizados no desenvolvimento do presente trabalho.

2.6.1 O Microcontrolador ESP-32-WROOM-32

O *ESP-32-WROOM-32* é um módulo compacto de alto desempenho que integra *Wi-Fi* e *Bluetooth* de baixo consumo de energia elétrica, baseado no microcontrolador *ESP-32*, destinado a aplicações embarcadas e dispositivos IoT.

O mesmo implementa o microprocessador Xtensa dual-core 32-bit LX6, que funciona na frequência de 240 MHz. Possui 448 KB de ROM, 520 KB de SRAM e 8 KB de SRAM no *Real Time Clock* (RTC). A Figura 1 ilustra o ESP-32-WROOM-32 utilizado.

Figura 1 – ESP-32-WROOM-32



Fonte: <https://www.robocore.net/wifi/esp32-wifi-bluetooth?srsltid=AfmBOopijGssLedSGs20K4Np2jGWbm_Xm7bdMHGN7Me7szKvlpjfqD-w>

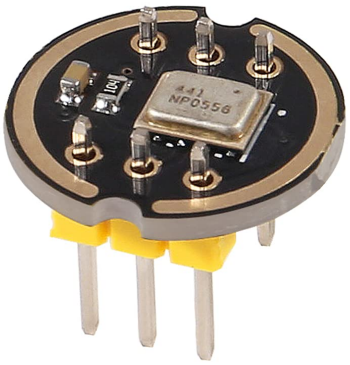
2.6.2 O Microfone INMP-441

O sensor INMP-441 é um microfone omnidirecional que apresenta saída de dados digital e baixo consumo energético. Com ele, é possível obter o som do ambiente em todas as direções.

O microfone opera com uma faixa de tensão de alimentação de 1,8 V a 3,3 V e uma

corrente de 1,4 mA. O mesmo possui uma Relação Sinal-Ruído (SNR) de 61 dBA. A Figura 2 ilustra o modelo de microfone utilizado.

Figura 2 – Microfone INMP-441

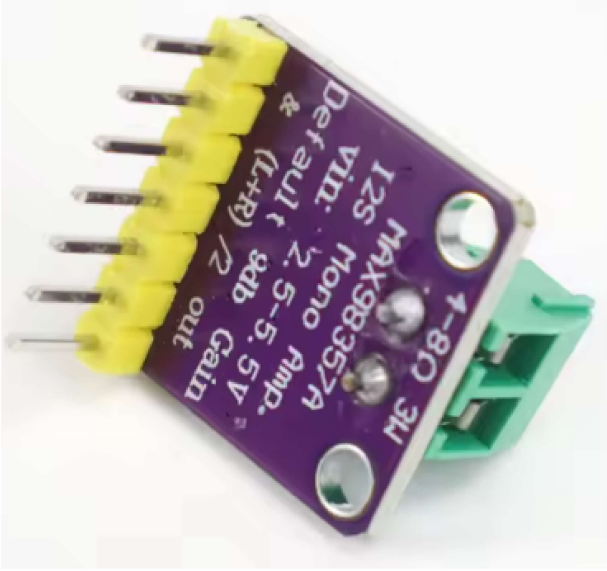


Fonte:
<https://www.amazon.com.br/ACEIRMC-Microfone-omnidirecional-INMP441-interface/dp/B0963PHKS7>

2.6.3 O Amplificador MAX-98357A

O MAX-98357A é um aplificador de sinal de áudio, que opera com tensões entre 2,5 V a 5,5 V à potência de 3 W. Neste projeto, ele será utilizado para auxiliar no controle do alto falante por meio do protocolo de comunicação serial I²S. A Figura 3 ilustra o modelo de amplificador utilizado.

Figura 3 – Amplificador MAX-98357A

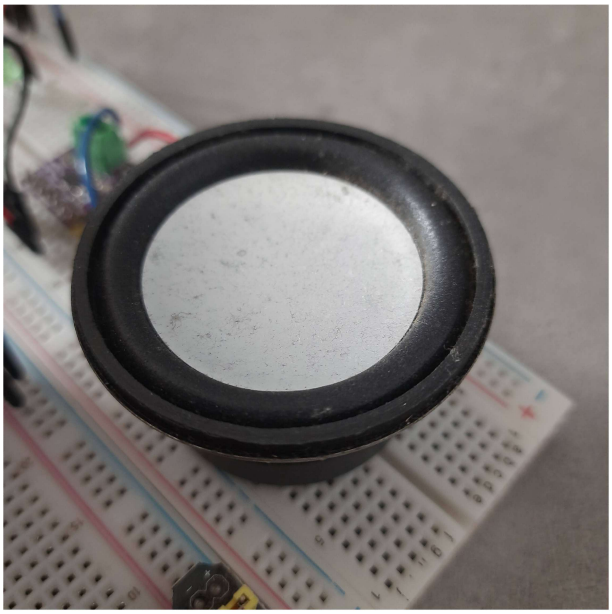


Fonte <https://pt.aliexpress.com/item/1005009355202579.html>

2.6.4 O Alto Falante

O alto falante será utilizado para a saída de som do sistema em questão. A Figura 4 ilustra o modelo de alto-falante utilizado. Sua impedância foi empiricamente medida e é de 4 Ω, sendo compatível com o amplificador que será utilizado neste projeto.

Figura 4 – Alto falante utilizado neste projeto



Fonte: Autor

2.7 Métricas de Avaliação

A validação da eficácia do modelo de classificação proposto fundamenta-se na análise quantitativa de seu desempenho sobre o conjunto de testes. Para garantir que o assistente EVA interprete os comandos com a confiabilidade necessária para um ambiente de automação, foram adotadas métricas estatísticas padrão na literatura de aprendizado de máquina.

A seguir, são detalhados os indicadores utilizados para mensurar a capacidade do modelo em distinguir corretamente as intenções (Verdadeiros Positivos) e sua robustez contra interpretações errôneas (Falsos Positivos e Falsos Negativos).

2.7.1 Precision

Para avaliar a precisão dos comandos em texto classificados, usa-se a seguinte fórmula:

$$Precision = \frac{TP}{TP + FP}. \tag{2.1}$$

Por definição, TP representa a quantidade de comandos devidamente classificados e FP, a quantidade dos falsos-positivos. Com esta métrica, a precisão será maior quando houver resultados com menos falsos positivos.

2.7.2 Recall

Para a avaliação com o *Recall*, tem-se a seguinte definição:

$$Recall = \frac{TP}{TP + FN}. \tag{2.2}$$

Esta equação é bastante parecida com a da precisão, o termo *FP* é substituído pelo termo *FN* no denominador, que representa os falsos-negativos. Dessa forma, o *recall* penaliza a quantidade de falsos-negativos. Ou seja, quanto menos falsos-negativos, maior será o *recall*.

2.7.3 F1-score

A métrica obtida pelo *F1-score* é uma soma harmônica entre *precision* e *recall*. Diante disso, sua definição é:

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.3}$$

Dessa forma, o *F1-score* é uma métrica balanceada que penaliza a variável que possuir menor pontuação. Logo, pode-se concluir que o *F1-score* é uma métrica balanceada.

2.7.4 Acurácia

A acurácia é uma das métricas mais comuns para avaliaro desempenho de modelos de *machine learning*. Refere-se à exatidão a qual um modelo estatístico prediz uma resposta ou se ajusta aos dados. O percentual de acertos é medido de acordo com a fórmula 5.4:

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.4}$$

Onde:

- Verdadeiro Positivo (VP): quantidade de amostras que a classe é predita como "positiva"e de maneira adequada pelo modelo.
- Verdadeiro Negativo (VN): quantidade de amostras que a classe é predita como "negativa"e de maneira adequada pelo modelo.
- Falso Positivo (FP): quantidade de amostras que a classe é predita como "positiva"e de forma falsa pelo modelo.
- Falso Negativo (FN) quantidade de amostras que a classe é predita como "negativa"e de forma falsa pelo modelo.

2.7.5 Matriz de Confusão

A matriz de confusão constitui uma ferramenta fundamental para a análise aprofundada do desempenho de algoritmos de classificação. Ela permite visualizar as proporções de acertos e erros entre as diferentes classes, servindo como base para o cálculo de diversas métricas de avaliação. Essa matriz apresenta as quantidades de Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN). No caso de classificadores multiclasse, os resultados são organizados em uma matriz de confusão de dimensão $k \times k$, em que k corresponde ao número de classes. A figura 5 representa uma matriz de confusão 2×2.

Figura 5 – Matriz de confusão.

		VALOR PREDITO	
		SIM	**NÃO**
REAL	**SIM**	VERDADEIRO POSITIVO (VP)	VERDADEIRO NEGATIVO (VN)
	NÃO	FALSO POSITIVO (FP)	FALSO NEGATIVO (FN)

Fonte: autoria própria.

3 TRABALHOS RELACIONADOS

Neste capítulo, alguns trabalhos que servirão de base para o projeto em questão serão apresentados.

3.1 Assistente Virtual Inteligente Sabe de Sua Vida

A literatura apresenta diversas abordagens para implementação de assistentes virtuais. O trabalho de Chung e Lee (2018). debate sobre uma arquitetura baseada em microprocessadores e computação em nuvem, presente nos dispositivos da marca *Amazon*, como o *Echo Dot* e *Dash Wand*, que implementam o funcionamento da Assistente Virtual Inteligente (AVI) Alexa para realizar operações simples do cotidiano a partir de comandos de voz, como tocar uma música ou criar uma lista de tarefas, dependendo principalmente de conexão com a *Internet* para conseguir processar os dados obtidos do usuário e realizar as tarefas previamente solicitadas.

Para Chung e Lee (2018), uma AVE pode representar um problema de privacidade ao usuário final, visto que apesar de todas as formas de tentar manter o sigilo das informações antes e depois de serem enviadas e recebidas pelo servidor, de modo que os dispositivos funcionem sem informar diretamente ao servidor as informações do usuário (como criptografar e enviar separadamente os dados), pode não ser suficiente, pois ainda há a possibilidade de falhas de segurança estarem presentes no sistema de comunicação entre uma AVE e os servidores remotos. Dessa forma, ainda há a possibilidade de *hackers* obterem acesso às informações dos usuários, bem como os padrões de comportamento e o histórico detalhado da atividade dos mesmos.

Contudo, criminosos podem se apropriar de informações interceptadas de usuários para monitorar a sua rotina, a exemplo: a localização e o horário de acordar e dormir. De acordo com Chung e Lee (2018), em um pior cenário possível, um *hacker* mal intencionado pode monitorar esses padrões de comportamento e se apropriar desses dados obtidos para planejar e realizar um roubo de pertences da pessoa monitorada.

3.2 Um Assistente Virtual com Habilitação de Fala para Interação Humano-Robô Eficiente em Ambientes Industriais

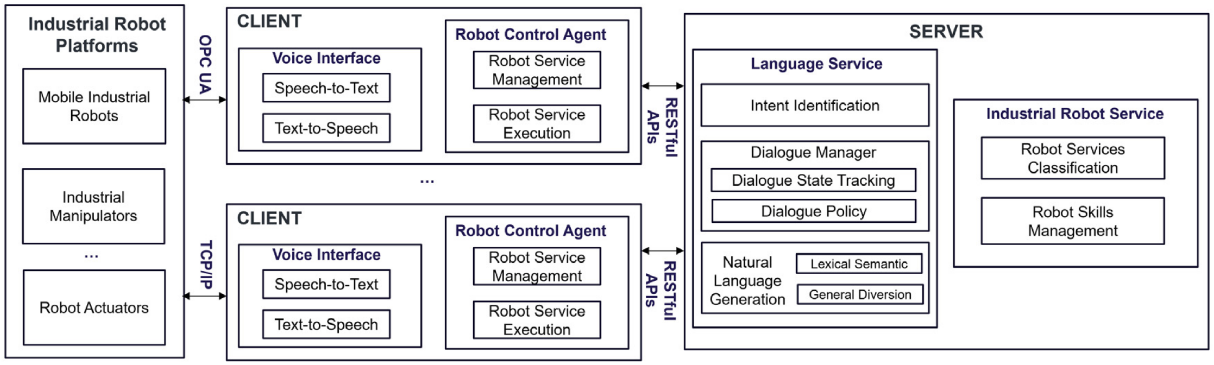
Li *et al.* (2023) desenvolveram um sistema denominado *Max*, um AV que apresenta uma interface de comunicação baseada em linguagem natural para facilitar a interação entre humanos e robôs industriais. O objetivo do sistema é otimizar operações no chão de fábrica,

permitindo a execução de tarefas como a logística interna (entrega de pacotes) e o mapeamento digital do ambiente (atualização de posições de interesse no mapa do robô).

A arquitetura do projeto Max é baseada no estilo Cliente-Servidor e organiza-se em três componentes principais: Plataformas Robóticas Industriais, Cliente e Servidor.

A representação das três camadas principais da arquitetura deste projeto pode ser visualizada na figura 6.

Figura 6 – Arquitetura geral HRI.



Fonte: Li *et al.* (2023).

3.2.1 Plataformas Robóticas Industriais

Esta categoria engloba os robôs móveis autônomos e manipuladores industriais presentes no ambiente de trabalho. A comunicação entre estas plataformas e o assistente virtual ocorre por meio de protocolos industriais padrão, como TCP/IP e OPC UA. Estes robôs são os responsáveis pela execução física das tarefas solicitadas verbalmente pelo operador.

3.2.2 Cliente (Max Client)

O cliente atua como o intermediário direto com o usuário e o controlador dos robôs. Ele é composto por uma Interface de Voz, responsável pelos processos de transcrição de fala para texto (Speech-to-Text) e síntese de texto para fala (Text-to-Speech). Além disso, o cliente hospeda o Agente de Controle do Robô, que recebe as instruções processadas pelo servidor e envia os comandos de execução diretamente para as plataformas robóticas.

3.2.3 Servidor (Max Server)

O servidor concentra a inteligência do sistema, implementando algoritmos de Processamento de Linguagem Natural (PLN). Ele utiliza o modelo BERT (*Bidirectional Encoder*

Representations from Transformers) ajustado (*fine-tuned*) para identificar a intenção do usuário e extrair parâmetros (*slots*) a partir do texto transcrito. O servidor também gerencia o diálogo e seleciona os serviços robóticos adequados, retornando ao cliente a resposta estruturada para que este possa comandar o robô e dar *feedback* verbal ao usuário.

3.3 Assistentes Virtuais na Indústria 4.0: Uma Revisão Literária Sistemática

Os autores Pereira *et al.* (2023) apresentam uma revisão sistemática da literatura sobre o uso de assistentes virtuais no contexto da Indústria 4.0, guiada pelo Princípio de Assistência Técnica (Hermann et al., 2016).

Os autores adotam a metodologia PRISMA para conduzir buscas em bases como ACM, IEEE e *ScienceDirect*, restringindo a seleção a estudos publicados entre 2018 e 2022 que explorassem soluções de assistentes virtuais ou *chatbots* aplicados à manufatura inteligente. Depois de filtrar centenas de artigos, eles chegaram a um conjunto de apenas oito estudos que detalham tanto interfaces conversacionais baseadas em linguagem natural quanto integrações com hardware de chão de fábrica, como robôs móveis e sistemas de realidade aumentada.

As soluções analisadas variam desde *chatbots* capazes de consultar em tempo real o estado de máquinas e indicadores de produção até protótipos de robôs equipados com sensores e atuadores que oferecem instruções visuais e alertas diretamente aos operadores. Apesar do potencial de acelerar processos de treinamento e apoiar decisões de operação, esses trabalhos enfrentam barreiras significativas, como a dificuldade de reconhecimento de voz em ambientes ruidosos, a complexidade de integrar múltiplos sistemas legados e a quase total ausência de mecanismos robustos de segurança e privacidade dos dados industriais.

Por fim, os autores destacam que o mercado global de assistentes virtuais tem crescido rapidamente, mas a aplicação prática na manufatura permanece escassa e fragmentada; sugerem, portanto, que pesquisas futuras concentrem-se em padronizar terminologias, explorar o uso de grandes modelos de linguagem como o GPT-4 e desenvolver interfaces de voz capazes de operar com baixa latência em condições adversas, além de combinar assistência virtual e física em arquiteturas ciber-físicas mais integradas.

3.4 Análise com assistentes virtuais inteligentes: Um estudo de caso com o Google Assistant

No estudo conduzido por Matos e Oliveira (2021), investiga-se o uso do *Google Assistant* por estudantes de Tecnologia em Sistemas para Internet em tarefas acadêmicas cotidianas, especialmente durante o Ensino Remoto Emergencial.

A pesquisa de abordagem mista incluiu dez alunos de diferentes semestres que, em sessões remotas, executaram nove tarefas típicas — desde a abertura de aplicativos e pesquisas simples até o download e salvamento de arquivos em nuvem, redação de documentos e envio de e-mails. Cada interação foi gravada, medida em tempo de execução, número de tentativas e taxa de sucesso, além de coletar impressões qualitativas sobre usabilidade e experiência. Os resultados revelam que, embora comandos isolados como "abrir o *Google Docs*" ou "pesquisar artigos" apresentem taxa de conclusão próxima a noventa por cento e sejam realizados em poucos segundos, tarefas mais complexas envolvendo navegação de arquivos, ditado de textos ou preenchimento de formulários falham em até setenta por cento dos casos, demandando várias tentativas, gerando frustração e perda de confiança no assistente.

Matos e Oliveira (2021) apontam ainda que o *Google Assistant* em português não mantém contexto de diálogo e exige a invocação repetida do comando de ativação, o que torna os fluxos de trabalho longos e desconfortáveis. A partir do estudo de Matos e Oliveira (2021), conclui-se que, embora promissor para consultas rápidas, o assistente de voz atualmente não atende às demandas acadêmicas que exigem precisão e continuidade de diálogo; Matos e Oliveira (2021) recomenda o desenvolvimento de assistentes virtuais pessoais com melhor reconhecimento de voz em português, capacidade de gerenciamento de contexto e integração híbrida de entradas por voz e toque, além de novas pesquisas comparando diferentes plataformas e explorando modelos de linguagem avançados para aprimorar a interação feita entre o usuário e o assistente virtual.

3.5 Assistente Virtual Baseada em IA usando Python

O artigo *"AI-Based Virtual Assistant Using Python: A Systematic Review"* analisa de forma abrangente o desenvolvimento de assistentes virtuais baseados em inteligência artificial implementados em Python, destacando sua evolução histórica, arquitetura de sistema e potencial de aplicação em *desktops* Windows. Inicialmente, os autores definem assistente virtual como um

agente de software capaz de executar tarefas por comando de voz ou texto e percorrem marcos que vão desde sistemas iniciais de reconhecimento de fala até as plataformas modernas, como Alexa e Siri, enfatizando o papel dos módulos de reconhecimento automático de fala (ASR) e de processamento de linguagem natural (PLN) no mapeamento de comandos para funções executáveis. Em seguida, apresentam uma proposta de arquitetura conceitual que detalha fluxos de dados em três etapas principais — captura de áudio, interpretação por meio de APIs web via HTTP e exibição de resultados em navegador — evidenciada por diagramas de fluxo que ilustram a comunicação entre *frontend* de reconhecimento de voz e *backend*.

No corpo do trabalho, os autores descrevem a integração de bibliotecas-chave do ecossistema Python, como *pyttsx3* para síntese e reconhecimento de voz, módulos de acesso ao sistema operacional, extração de dados, e pacotes de suporte a funcionalidades diversas — de consulta à Wikipédia a envio de e-mails e reprodução de mídia. Ressaltam que o sistema opera sob três modalidades de aprendizado de máquina (supervisionado, não supervisionado e por reforço), embora os experimentos reportados concentrem-se em tarefas básicas de automação de rotina, nas quais o assistente demonstra ganhos expressivos de produtividade ao poupar interações manuais e operar ininterruptamente.

Por fim, concluem que, embora seu protótipo consolide com sucesso funções elementares — como consultar o clima, abrir programas e executar pesquisas —, ainda há espaço para ampliar sua robustez, migrando de soluções puramente baseadas em regras para arquiteturas híbridas que incorporem modelos de linguagem profundas e suportem diálogos contextuais mais sofisticados, além de explorar aplicações em IoT e integração com dispositivos móveis.

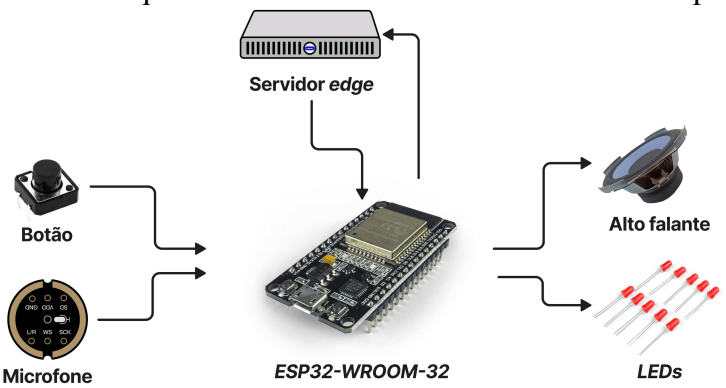
4 ARQUITETURA DO SISTEMA PROPOSTO

A arquitetura proposta para implementar o sistema da EVA segue o paradigma de computação distribuída em borda, onde o processamento é dividido entre dois níveis hierárquicos. Este modelo Cliente-Servidor, análogo ao proposto por Li *et al.* (2023) para ambientes industriais, separa a interface com o usuário e o controle de atuadores (cliente) do processamento de linguagem natural (servidor). Essa separação permite otimizar o consumo energético, mantendo capacidade de processamento adequada para tarefas complexas de IA e, crucialmente, mitiga os riscos de privacidade associados a arquiteturas puramente baseadas em nuvem, como as analisadas por Chung e Lee (2018).

4.1 Visão Geral do Sistema Proposto

O sistema irá coletar informações sobre o ambiente por meio das entradas de dados (botão e microfone) como mostrado na figura 7. Os dados coletados serão enviados para a aplicação no ESP-32 onde ocorrerá o processamento de tarefas simples, como a verificação da interrupção do botão e captação do sinal de áudio do microfone, imediatamente os dados serão enviados para um servidor na edge por meio do protocolo TCP/IP, onde ocorrerá o processamento de tarefas mais complexas, como a interpretação dos comandos de voz e conversão de texto para fala por meio de Redes Neurais Artificiais, os dados processados pelo servidor retornarão ao microcontrolador no formato de JSON, onde eles serão interpretados e devidamente direcionados para as saídas de dados (LEDs e alto falante), a fim de retornar um *feedback* visual e/ou auditivo para o usuário.

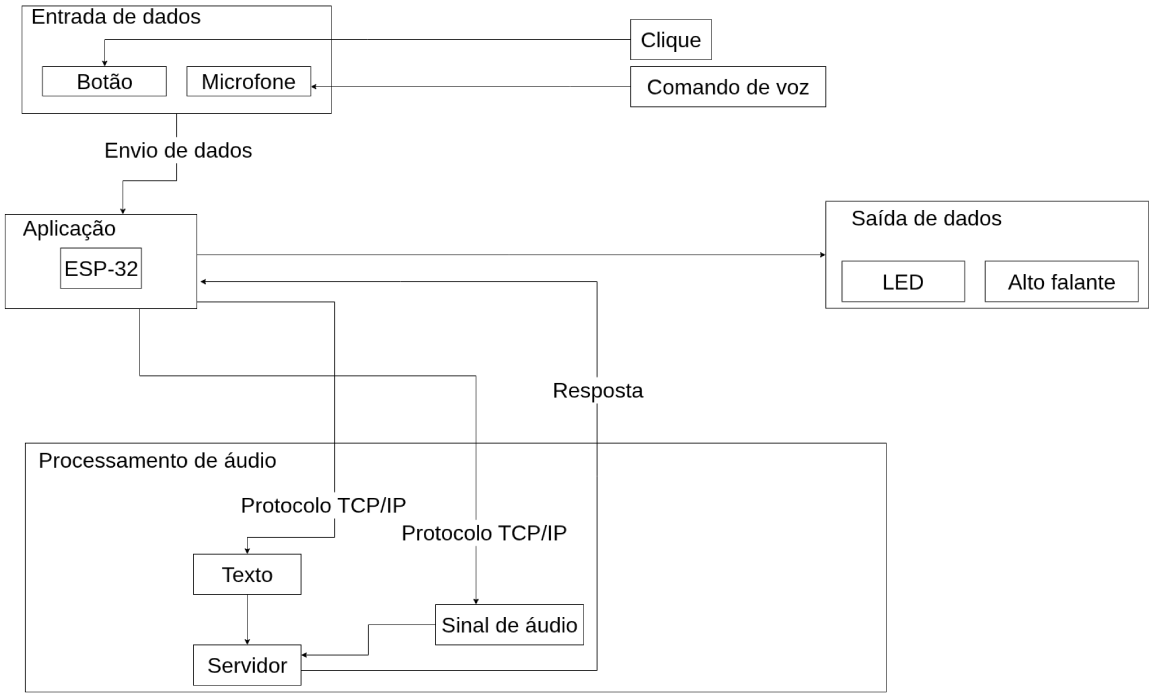
Figura 7 – Arquitetura de Funcionamento do Sistema Proposto.



Fonte: autoria própria.

O funcionamento do projeto pode ser melhor entendido através do fluxograma mostrado na figura 8.

Figura 8 – Fluxograma geral do projeto EVA.



Fonte: autoria própria.

4.2 Módulo de Aquisição (ESP-32)

O módulo de aquisição é responsável por:

- Captura de áudio através do microfone INMP441 com taxa de amostragem de 16 kHz
- Iniciar a gravação de áudio ao pressionar o botão
- Reprodução de *feedback* sonoro via alto-falante
- Interface com sensores e atuadores locais
- Comunicação com módulo de processamento via Wi-Fi

O ESP-32 opera em modo de baixo consumo, ativando-se apenas quando o botão para gravar o comando de voz é pressionado. Ele verifica o status da conexão e realiza a troca de dados com o servidor de borda. A implementação utiliza FreeRTOS para gerenciamento de tarefas concorrentes.

4.3 Módulo de Processamento

O servidor em borda é constituido pelas seguintes especificações de *hardware*: GeForce GT 730 com 4 GB de RAM, Intel® Pentium® G4400 de dois núcleos, com a frequência de *clock* de 3,3 GHz, 16 GB de memória RAM DDR3 a 1333 MHz e tendo como sistema operacional Linux na distribuição Ubuntu, na versão 25.04 de 64 bits.

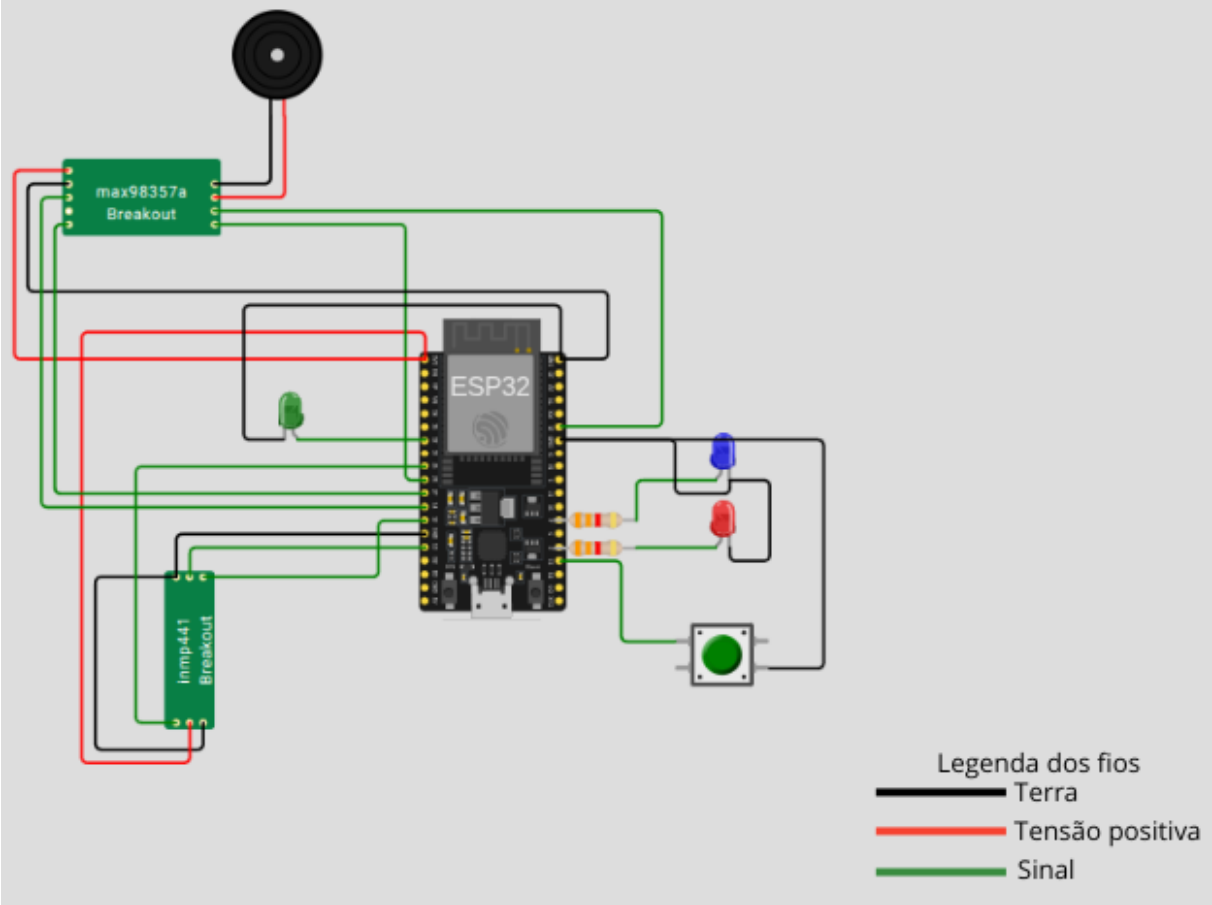
O servidor local executará as tarefas computacionalmente intensivas do projeto EVA, sendo estas:

1. Transcrição de áudio para texto usando framework de rede neural artificial Vosk otimizado para obter respostas rápidas
2. Processamento de linguagem natural para identificação de sensores e/ou atuadores mencionadas
3. Sistema de tomada de decisão baseado em redes neurais artificiais
4. Gerenciamento de estado dos dispositivos conectados

4.4 Diagrama de Blocos

O projeto do *hardware* usado para representar o projeto EVA, juntamente com os fios, sensores e atuadores acoplados ao sistema se encontra no diagrama de conexões da figura 9.

Figura 9 – Diagrama de blocos



Fonte: autoria própria.

5 METODOLOGIA

Nesta sessão apresenta-se o processo metodológico para definir os padrões de projeto a seguir para escolher os sensores, atuadores, microcontrolador e linguagens de programação a serem utilizados. O desenvolvimento deste projeto baseou-se na ordem de pesquisa a seguir:

1. Estudo bibliográfico: nesta etapa foi feita uma busca e revisão da literatura acadêmica sobre os modelos e/ou frameworks de Redes Neurais Artificiais para processamento de sinais de áudio, com o propósito de avaliar suas relevâncias, eficácias e metodologias.
2. Análise de projetos similares: Neste estágio do trabalho, foram realizadas comparações entre modelos e *frameworks* diferentes, a fim de determinar qual modelo teria o melhor desempenho e acoplamento ao projeto. Esta análise foi guiada por desafios conhecidos na literatura, como a dificuldade de reconhecimento de voz em ambientes com ruído e a necessidade de segurança de dados industriais, pontos levantados por (PEREIRA *et al.*, 2023), bem como as falhas de manutenção de contexto de diálogo observadas por (MATOS; OLIVEIRA, 2021) em assistentes comerciais.
3. Implementação do protótipo e validação dos testes: nesta fase foram realizados testes manuais, validando se o servidor *edge* consegue interpretar os comandos de voz de maneira adequada e se o microcontrolador é capaz de realizar a tomada de ações de maneira devida, sendo consistente e coeso com a solicitação do usuário. A validação focará em métricas de latência e taxa de sucesso, similarmente à abordagem de (MATOS; OLIVEIRA, 2021), e na robustez da comunicação, um fator crítico para a confiabilidade de sistemas de assistência física e virtual.

Para a parte de *hardware*, foram escolhidos o microcontrolador ESP-32-WROOM-32, microfone INMP-441, o amplificador MAX-98357A, a fonte HW-131, um *push button* e um alto falante.

Na parte de *software*, foram usadas as linguagens de programação C++ e *Python*, para programar o microcontrolador e servidor, respectivamente. O ambiente de desenvolvimento escolhido é o *Visual Studio Code*, pela variedade e suporte de extensões que facilitam iniciar e prosseguir com os projetos sem trocar de editor de código.

5.1 Seleção da Base de Dados

Esta sessão aborda como as bases de dados para processar o sinal de áudio e processar o texto foram definidas.

Para que o servidor local consiga processar o texto transcrito do áudio que o usuário enviará, foi definida uma base de dados padrão compatível com as ações dos periféricos acoplados ao projeto EVA.

A tabela 1 define a estrutura da base de dados definida para este projeto.

Tabela 1 – Descrição dos campos do arquivo de configuração NLU

Campo (Chave)	Tipo	Descrição
`text_examples`	Array	Lista principal contendo os objetos de treinamento de frases.
`.text`	String	A frase de exemplo como seria transcrita à partir do sinal de áudio.
`.intent`	String	O rótulo da intenção que o modelo deve classificar (ex: contar piada, controlar um atuador etc).
`.action`	String	O nome da função ou ação que o sistema deve executar em resposta.
`.entities`	Array	Lista de identificadores ou valores para entidades presentes na frase (ex: *lights, speaker, bulb* etc).
`entities_examples`	Object	Objeto que agrupa as categorias de entidades e seus sinônimos.
`.[categoria]`	Array	Lista de strings contendo variações léxicas (sinônimos) para a entidade (ex: light, lamp, bulb).

Fonte: Elaborada pelo autor (2025).

O quadro 5.1 exemplifica o formato de JSON utilizado para treinar a rede neural para realizar o PNL.

Quadro 5.1 – Exemplo de estrutura JSON para treinamento de NLU

```
{
  "text_examples": [
    {
      "text": "hit me with a joke",
      "intent": "tell_joke",
      "action": "speak_joke",
      "entities": [ 0, "null" ]
    },
    {
      "text": "tell me something hilarious",
      "intent": "tell_joke",
      "action": "speak_joke",
      "entities": [ 0, "null" ]
    }
  ],
  "entities_examples": {
    "light": [
      "light", "lights", "lamp",
      "lamps", "bulb", "bulbs"
    ],
    "led": [
      "led", "leds", "strip", "strips"
    ],
    "temperature_sensor": [
      "temperature", "sensor",
      "thermometer", "thermostat",
      "temp", "heat"
    ]
  }
}
```

Fonte: Elaborado pelo autor (2025).

5.2 Classificação

Nesta etapa, o sistema emprega um algoritmo de aprendizado supervisionado para categorizar as sentenças transcritas em intenções específicas. O modelo escolhido para o processamento de linguagem natural (PLN) do projeto EVA foi o *Multilayer Perceptron* (MLP), uma arquitetura clássica de Redes Neurais Artificiais do tipo *feedforward*.

A escolha do MLP justifica-se pela sua capacidade de aproximação universal e eficiência em classificar padrões não lineares complexos, características essenciais para interpretar as variações léxicas presentes nos comandos de voz do usuário. O algoritmo recebe como entrada os vetores representativos das frases (processados a partir da base de dados JSON apresentada na Quadro 5.1) e, através de camadas ocultas de neurônios, calcula a probabilidade de a frase pertencer a uma das intenções mapeadas (ex: `tell_joke`, `turn_on_light`), ativando a ação correspondente no microcontrolador.

6 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos à partir dos testes aplicados aos modelos de rede neural do projeto EVA para processar texto. Adicionalmente, serão expostos exemplos do funcionamento da parte de hardware do projeto.

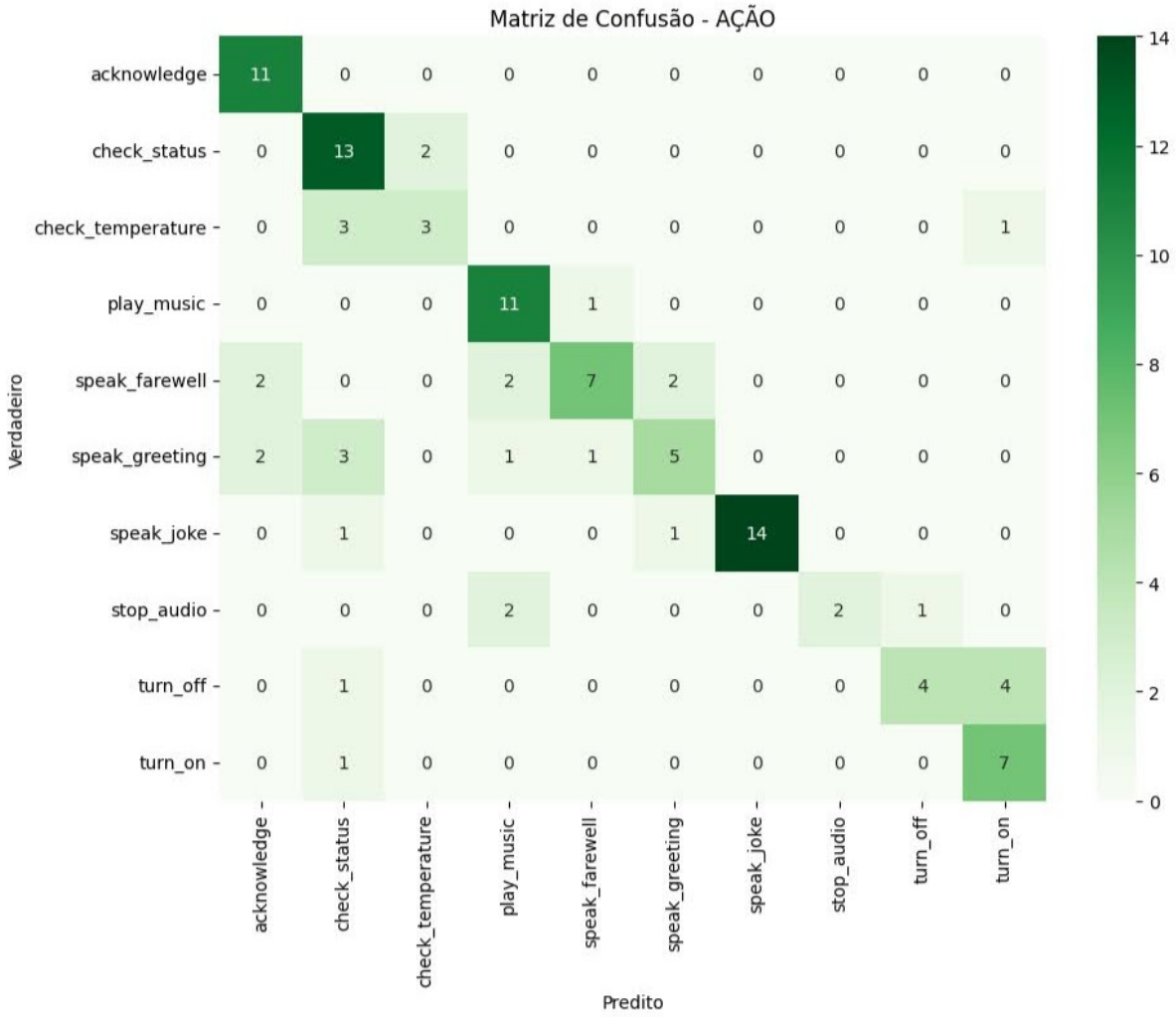
6.1 Resultados experimentais

Nessa seção será abordada os resultados do treinamento da rede neural, bem como a exposição das métricas usadas para avaliar os modelos e as convergências.

6.1.1 Matrizes de confusão

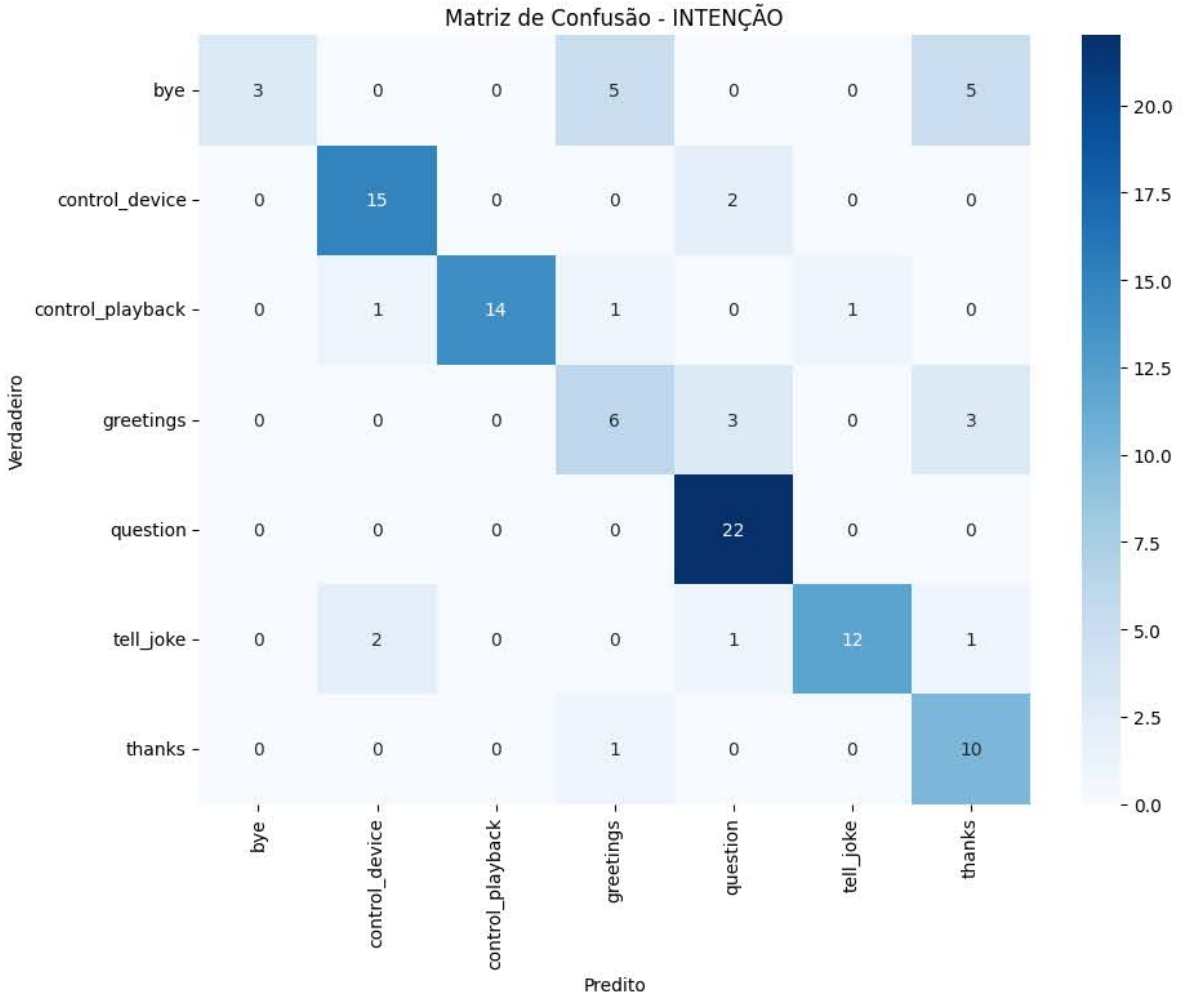
Nesta subseção serão expostas as matrizes de confusão dos modelos de processamento de texto para classificar as ações, que está presente na figura 10 e intenções, encontrada na figura 11.

Figura 10 – Matriz de confusão do modelo para a classe ação.



Fonte: autoria própria.

Figura 11 – Matriz de confusão do modelo para a classe intenção.



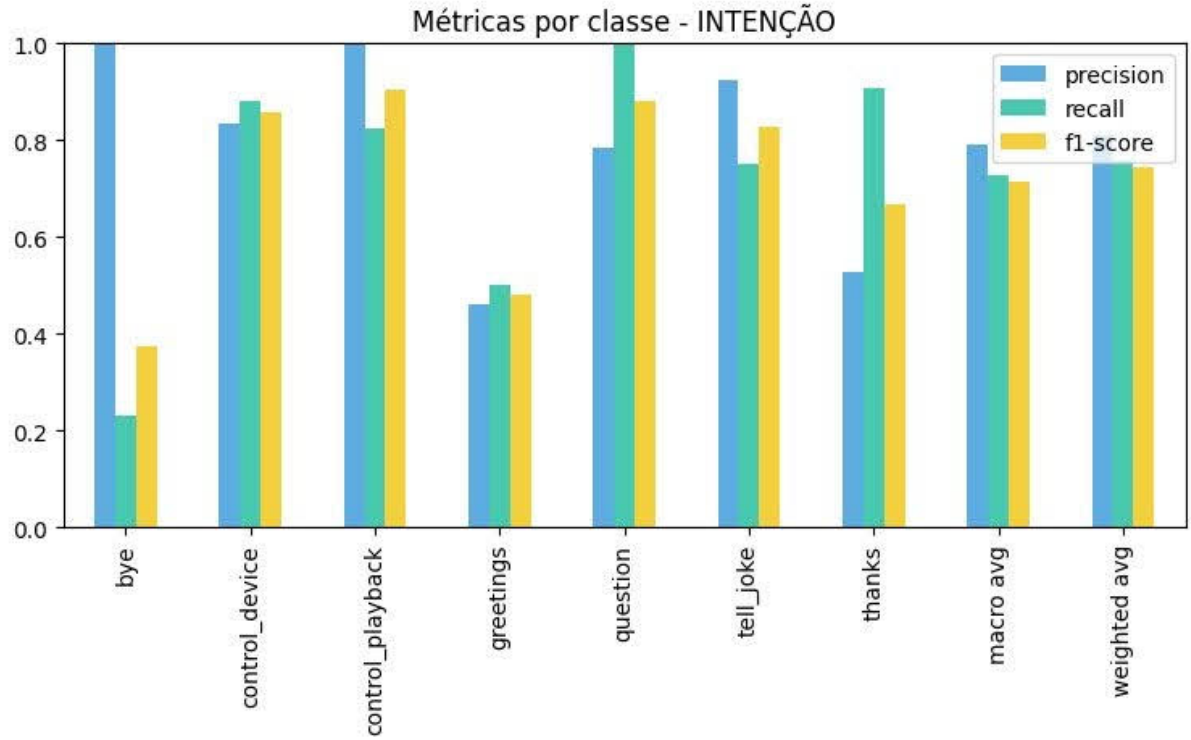
Fonte: autoria própria.

6.1.2 Desempenho dos modelos

No treinamento para ambos os modelos foram usadas uma taxa de 70% para treinamento e 30% para testes. O resultado da precisão obtida foi de 75,93% para o modelo de classificação de intenção e quanto ao de ação, 71,3%.

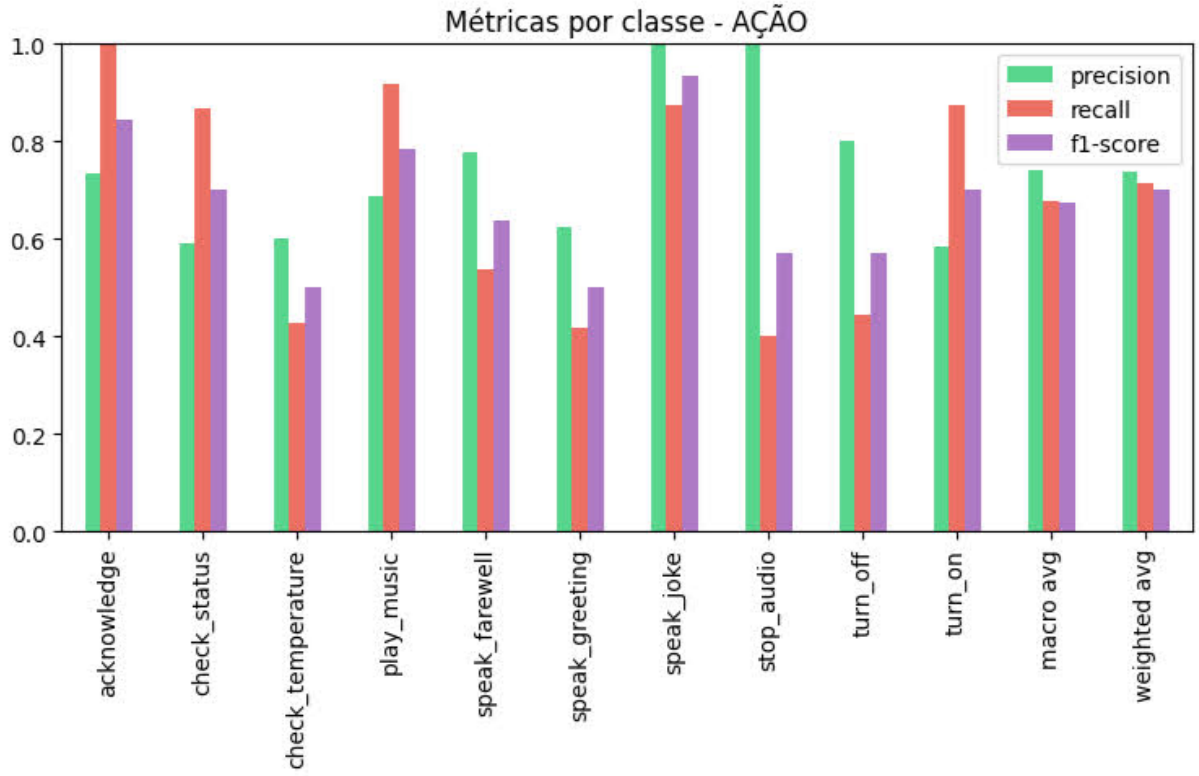
Nesta subseção também serão mostradas as pontuações obtidas por meio das métricas *precision, recall* e *f1-score* para classificar as intenções, que se encontra na figura 12 e ações, presente na figura 13.

Figura 12 – Métricas para a classe intenção.



Fonte: autoria própria.

Figura 13 – Métricas para a classe ação.

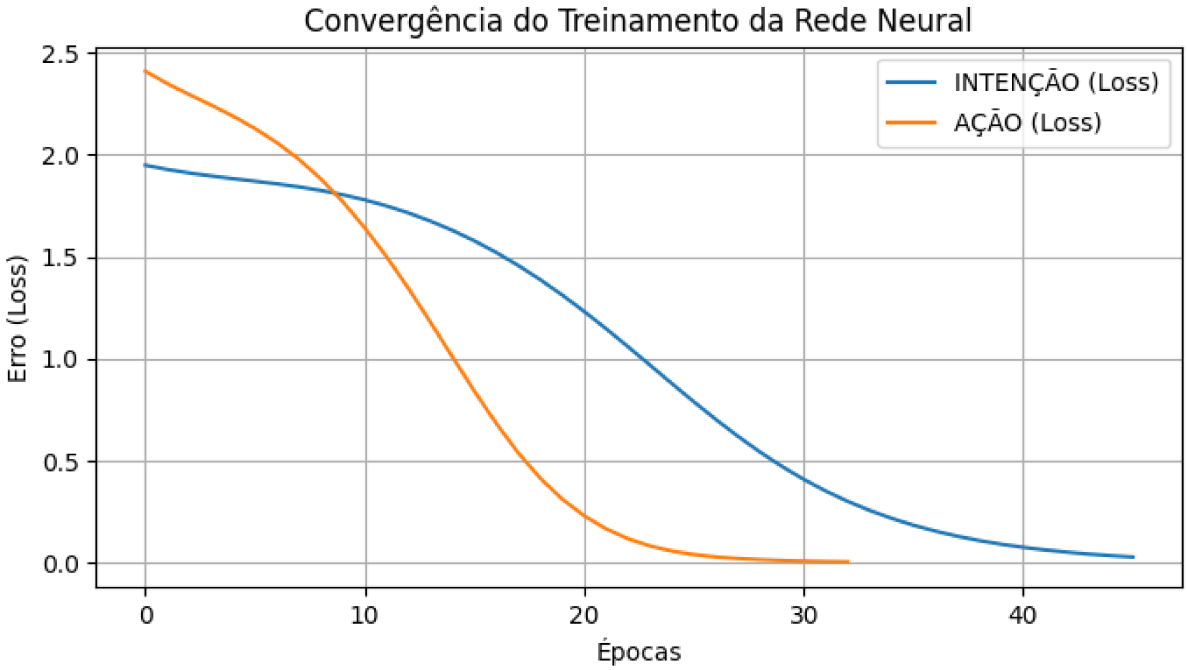


Fonte: autoria própria.

6.1.3 Convergência do modelo

Pode-se notar que na figura 14, o modelo para classificar as ações obtém uma taxa de erro consideravelmente baixa à partir de 20 épocas. Enquanto o modelo de classificação de ação, entre 30 e 40 épocas.

Figura 14 – Matriz de confusão para a convergência dos treinamentos dos modelos para intenção e ação.



Fonte: autoria própria.

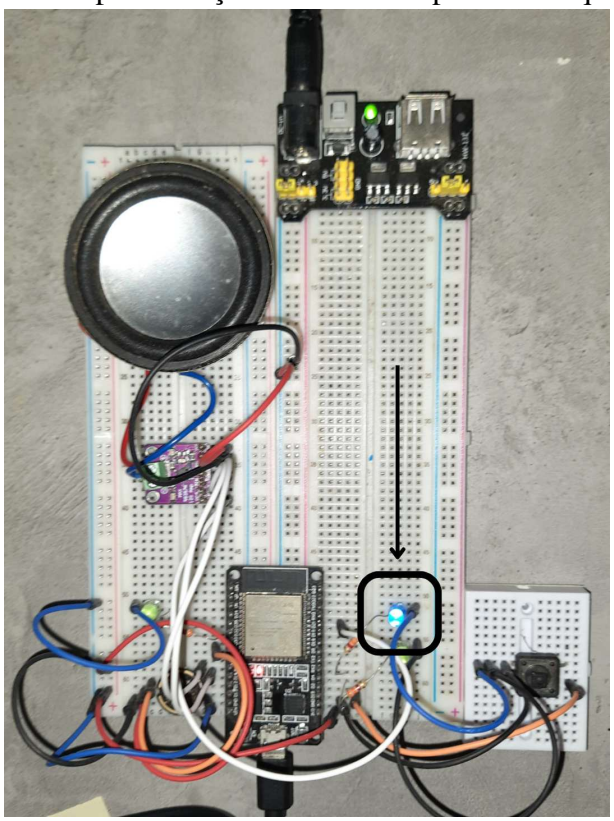
6.2 Resultados obtidos

Nesta seção será falado sobre alguns resultados obtidos de maneira experimental, no funcionamento do projeto na parte de hardware e software.

6.2.1 Funcionamento no hardware

A figura 15 representa a implementação do circuito do sistema EVA no mundo real. Pode-se perceber que o LED azul está aceso, indicando que a conexão com o servidor *edge* foi corretamente estabelecida.

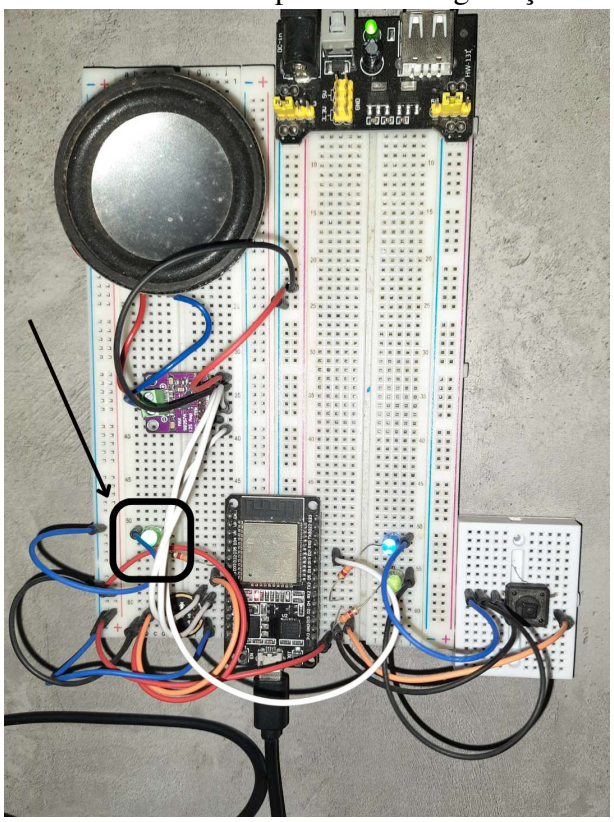
Figura 15 – Implementação do circuito à partir do esquemático.



Fonte: autoria própria.

Na figura 16 representa o LED do canto superior esquerdo aceso, indicando que o microfone foi acionado e está corretamente gravando o áudio do ambiente e enviando os dados para o servidor local.

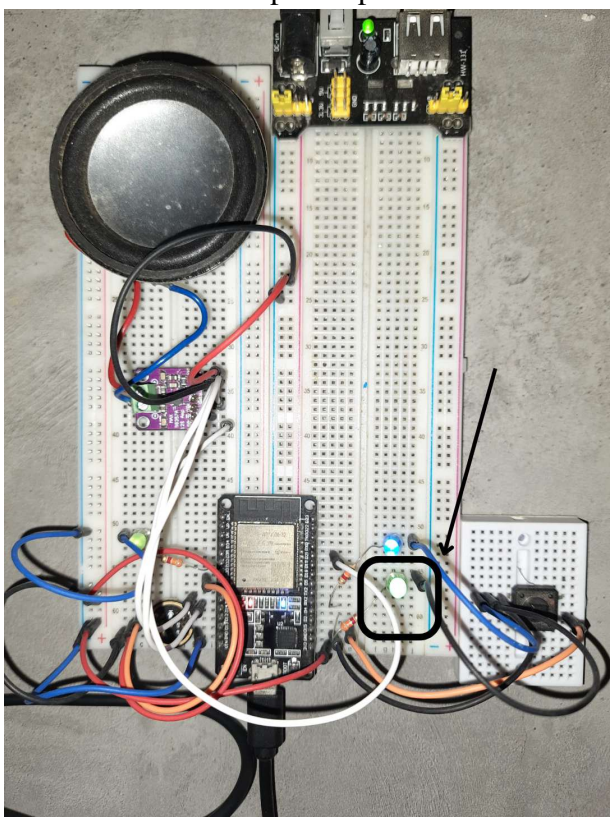
Figura 16 – Circuito com o LED para sinalizar gravação de áudio aceso.



Fonte: autoria própria.

Por fim, na figura 17, percebe-se que o outro LED verde está aceso, isto indica que o estado de variável “luz da cozinha” (que foi ficticiamente determinada no código) possui valor verdadeiro. Caso contrário, o mesmo LED estaria desligado.

Figura 17 – Circuito com o LED para representar a luz da cozinha aceso.



Fonte: autoria própria.

O relatório gerado pelo servidor local está presente na imagem abaixo. Pode-se perceber que o último comando *"turn the lights on"* foi devidamente recebido e processado pela aplicação *backend*.

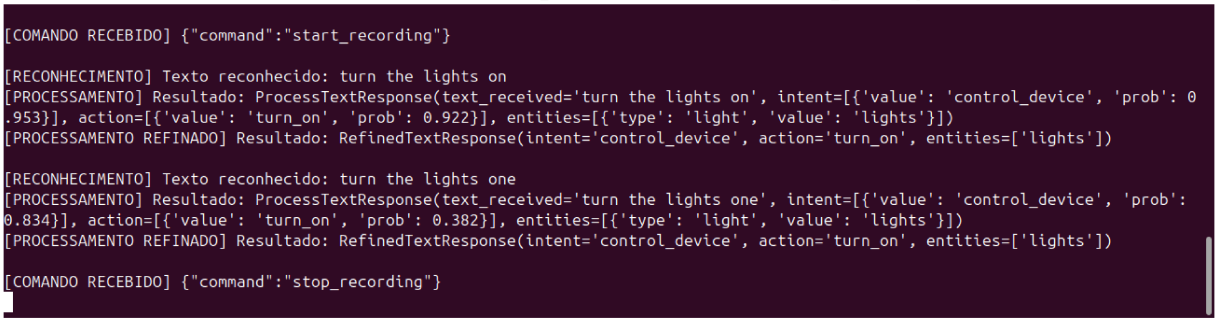
6.2.2 Funcionamento no software

Nesta subseção, alguns dos resultados do funcionamento do servidor local serão mostrados.

A Figura 18 ilustra o servidor local recebendo os dados de áudio do ESP-32, onde o usuário solicita para ligar a luz. O sinal de áudio é processado e o resultado do processamento é mostrado na saída do *terminal*.

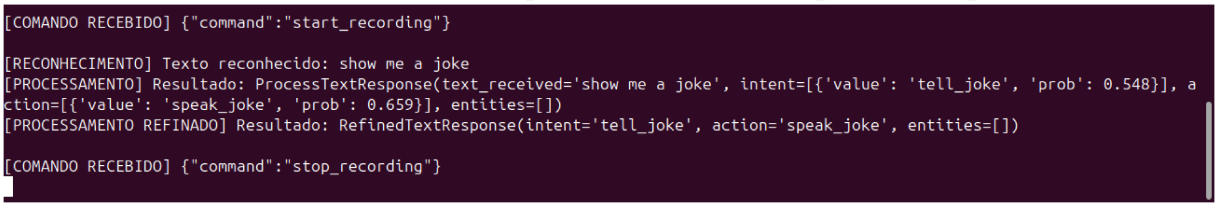
As próximas figuras representam o funcionamento do mesmo servidor local recebendo o áudio do usuário, processando-o em tempo real, mostrando o resultado da transcrição. Dessa vez, o usuário solicita para que a assistente conte uma piada na Figura 19 e para verificar o valor da temperatura na Figura 20.

Figura 18 – Servidor local processando comando para ligar a luz.



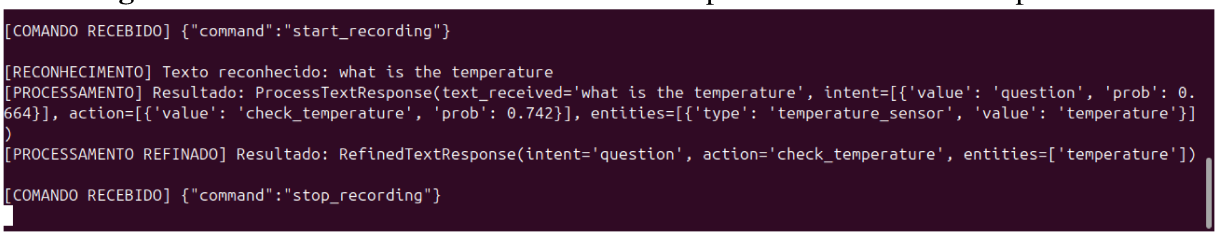
Fonte: autoria própria.

Figura 19 – Servidor local processando comando para contar piada.



Fonte: autoria própria.

Figura 20 – Servidor local recebendo comando para fazer leitura da temperatura.



Fonte: autoria própria.

6.3 Análise dos resultados

Os modelos apresentam acurácias acima de 70%. Onde, para as intenções, foram detectados 26 amostras como erro de um total de 108 amostras de testes, que presenta 24,1% de divergência. Quanto às ações, houve 31 previsões incorretas de um total de 108 amostras, resultando em 28,7% de divergência.

As matrizes de confusão e os relatórios de classificação (métricas por classe) fornecem uma visão mais detalhada dos acertos e erros. Eles mostram quais classes são mais frequentemente confundidas e o desempenho do modelo em termos de precisão, *recall* e *f1-score* para cada intenção e ação individualmente.

Nos testes de implementação no mundo real, o projeto tem-se mostrado com desempenho satisfatório e funcional, apesar de possuir várias limitações.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou a EVA, uma assistente virtual embarcada que implementa uma capacidade de processamento avançado através de arquitetura distribuída. Os resultados experimentais demonstram viabilidade técnica da solução proposta.

A separação entre aquisição de dados e processamento permite otimizar recursos enquanto mantém a funcionalidade. O foco em processamento local atende as demandas restritas e personalizadas para controlar um circuito de baixa complexidade, que está acoplado ao ESP-32.

A arquitetura modular facilita expansão e customização, tornando o sistema adequado para diferentes cenários de aplicação. Os protocolos de rede utilizados garantem operação confiável em ambientes conectados.

7.1 Trabalhos futuros

No sentido de dar continuidade a este projeto, é possível pensar em algumas abordagens a serem exploradas:

1. Implementar um algoritmo de criptografia para a troca de dados entre o ESP e o servidor.
2. Criar um *frontend* para controlar os atuadores do ESP.

REFERÊNCIAS

BENDEL, M. C. Edge computing: conceitos, arquiteturas e desafios. 2018. Disponível em: <https://www.maxwell.vrac.puc-rio.br/34297/34297.PDF>.

CHUNG, H.; LEE, S. **Intelligent Virtual Assistant knows Your Life**. Seoul, 2018. Disponível em: <https://doi.org/10.48550/arXiv.1803.00466>.

COUTINHO, B. L. Estudo de plataformas para computação distribuída e suas aplicações. 2021. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/226079/001140062.pdf>.

DIAS, J. F. C.; FILHO, J. D. L. Internet das coisas (IoT) e indústria 4.0: revolucionando o mundo dos negócios. **Revista Interface Tecnológica**, v. 15, n. 1, p. 71–81, 2018. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/download/496/300/2112>.

ESCOLA SUPERIOR DE REDES. **Arquitetura TCP/IP: conceitos básicos**. 2020. Site institucional. Disponível em: <https://esr.rnp.br/noticias/arquitetura-tcpip-conceitos-basicos/>.

ESPRESSIF SYSTEMS. **FreeRTOS (IDF): Symmetric Multiprocessing (SMP)**. Shanghai, 2024. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/freertos-smp.html>.

FORTINET. **O que é protocolo de controle de transmissão TCP/IP?** 2025. Site institucional. Disponível em: <https://www.fortinet.com/br/resources/cyberglossary/tcp-ip-protocol>.

GOMES, M. L. Computação em borda no contexto da internet das coisas: uma revisão sistemática. **Revista Brasileira de Computação Aplicada**, v. 15, n. 2, p. 50–65, 2023. Disponível em: <https://seer.upf.br/index.php/rbca/article/view/1545/1409>.

HE, N.; HUANG, H.-W. Use of freertos in teaching real-time embedded systems design course. **Minnesota State University, Mankato**, 2014. Paper ID #10607.

LI, C.; CHRYSOSTOMOU, D.; YANG, H. A speech-enabled virtual assistant for efficient human–robot interaction in industrial environments. **Journal of Systems & Software**, v. 205, p. 111818, 2023. Disponível em: <https://doi.org/10.1016/j.jss.2023.111818>.

MATOS, D. R. d. S.; OLIVEIRA, F. K. d. Análise com assistentes virtuais inteligentes: Um estudo de caso com o google assistente. **Revista Novas Tecnologias na Educação**, v. 19, n. 1, p. 473–483, jul 2021. Disponível em: <https://doi.org/10.22456/1679-1916.118537>.

PATIL, K. M. M.; PATIL, A.; SHINDE, S.; PATRA, S.; PATIL, S. Ai-based virtual assistant using python: A systematic review. **International Journal for Research in Applied Science & Engineering Technology (IJRASET)**, v. 11, n. 3, p. 814–818, mar 2023. Disponível em: <https://doi.org/10.22214/ijraset.2023.49519>.

PEREIRA, R.; LIMA, C.; PINTO, T.; REIS, A. Virtual assistants in industry 4.0: A systematic literature review. **Electronics**, v. 12, p. 4096, 2023. Disponível em: <https://doi.org/10.3390/electronics12194096>.

RIBEIRO, N. C. A importância da internet das coisas e da inteligência artificial na indústria 4.0. Juiz de Fora, 2023. Disponível em: <https://ri.unipac.br/repositorio/wp-content/uploads/tainacan-items/282/260222/NATHALIA-DE-CASSIA-RIBEIRO-A-IMPORTANCIA-DA-INTERNET-DAS-COISAS-ENGENHARIA-pdf>.

APÊNDICE A – CÓDIGOS

Os *scripts* foram implementados usando a plataformas de edição de código *Visual Studio Code*, na versão 1.106.3 e *Google Colab* versão *online*. As linguagens de programação utilizadas foram *Python*, na versão 3.13.3 e C++, utilizando o GCC como compilador na versão 14.2.0.

O repositório do *firmware* está disponível em <https://github.com/caioarodrigues/eva-firmware>, já o código do servidor local se encontra em <https://github.com/caioarodrigues/eva-ws> e por fim, o repositório da Rede Neural implementada está em <https://colab.research.google.com/drive/1aLSqr0jnofbcDiNhISjFxX5LNsa1db9V>.