UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Renato Augusto Schenkel Meneghin Marchiori

Mini Carro Robô Controlado por Voz

Florianópolis
2023

Renato Augusto Schenkel Meneghin Marchiori

Mini Carro Robô Controlado por Voz

Projeto Final da disciplina de Programação em Sistemas Embarcados - Relatório submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia Elétrica.
Orientador: Prof. Eduardo Augusto Bezerra

Florianópolis
2023

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este documento é um relatório do desenvolvimento do projeto proposto na disciplina de programação em Sistemas Embarcados. Ele se destina a codificar e montar um carro robô controlado por voz.

A pesquisa e o avanço nas redes neurais artificiais demonstraram a capacidade delas de trabalhar em diversas áreas, como processamento de imagens, som e vídeos (SOLOVYEV, R. *et al.*, 2019). Em diversas aplicações, demonstrou-se que as redes neurais profundas possuem uma capacidade melhor problemas de identificação e classificação que as redes neurais convencionais, e até mesmo especialistas humanos. As arquiteturas de redes neurais modernas incluem redes com camadas convolucionais, recebendo o nome de redes neurais convolucionais (CNN) (GALEA; CAPELO, 2018).

No contexto de reconhecimento de som, uma arquitetura de rede neural pode ser utilizada para aplicações embarcadas. Por exemplo, os telefones celulares modernos possuem assistentes móveis como o assistente Google, Alexa da Amazon ou a Siri da Apple (SOLOVYEV, R. A. *et al.*, 2020). No trabalho de Xiaoling (LV; ZHANG; LI, 2008), também é demonstrado que uma arquitetura redes neurais pode ser implementadas para realizar comandos de veículos. CNNs também foram implementadas em *Field Programable Gate Arrays* (FPGA) no conjunto de dados MNIST, obtendo um desempenho de 96% de acurácia.

Futuramente, haverá um crescimento exponencial das aplicações embarcada que podem agrupar não somente o reconhecimento por voz, mas também outras funcionalidades. Entretanto, para que o sistema seja funcional, ele deve ser capaz de realizar as operações matemáticas de inferência para funcionalidades em tempo real e para diversos tipos de sistemas. Pode-se ter também, considerações de memória disponível para armazenamento do modelo de rede neural.

Um projeto, então, foi apresentado para introduzir os alunos aos desafios da programação e de aplicações envolvendo microcontroladores. Os requisitos do projeto envolve um cenário com sensoriamento e atuação, implementado na linguagem C++, com capacidade suficiente para utilizar alguma modelagem de redes neurais pré-treinada e permitindo comunicação com um dispositivo hospedeiro, como um computador, via comunicação UART e comunicação sem fio.

De forma a cumprir esses requisitos, foi proposto a implementação de um mini carro robô controlado por voz. O robô deverá receber um comando de voz através do seu microfone, o qual usará uma modelagem de rede neural (por meio do TinyML) para auxiliar na interpretação do sinal recebido. Com o sinal decodificado, o robô acionará os respectivos motores para a sua atuação, por exemplo, ir para frente, para trás, girar para a esquerda ou para a direita. Toda vez que algum comando for executado, o robô guardará esse comando dentro de uma estrutura de armazenamento de dados

dinâmicos baseada em fila, junto com o horário do comando. Quando o dispositivo hospedeiro solicitar, o microcontrolador deverá transmitir a fila de todas as operações executadas e armazenar em uma lista.

1.1 MOTIVAÇÃO

A aplicação a qual o presente trabalho vislumbra é a aplicação desse classificador em comandos de máquinas. Robôs que se especializam em atividades específicas estão cada vez mais presentes e a introdução do comando por voz, pode auxiliar no comando e execução de suas tarefas, enquanto que o operador humano fica livre para executar outra (LV; ZHANG; LI, 2008). Ao identificar e classificar esses comandos gerados por voz, obtém-se mais um método de acesso, de operação e interação entre homem e máquina.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Implementar uma aplicação embarcada de um mini-robô controlado por comandos de voz

1.2.2 Objetivos Específicos

- Exercitar a modelagem OO visando implementação em C++;
- Entender os desafios do projeto integrado de software/hardware para sistemas embarcados;
- Praticar o fluxo completo de projeto de sistemas em C++ para sistemas embarcados;
- Exercitar a interação entre diferentes sistemas em software;
- Implementar um modelo de aprendizado de máquina no microcontrolador.

2 MODELAGEM

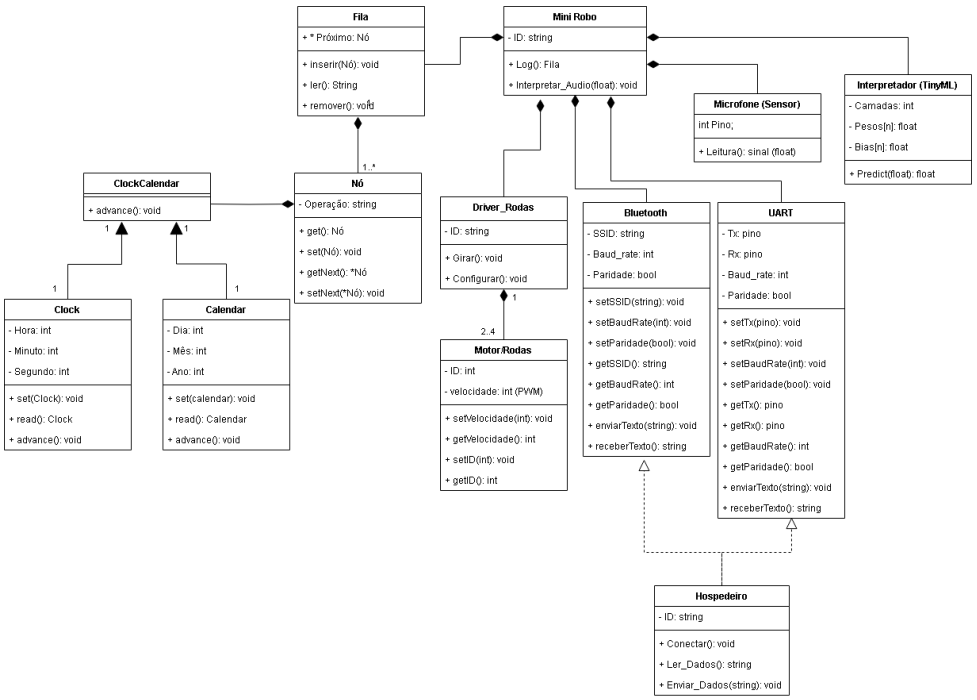
Neste capítulo serão discutidos os métodos considerações e outros aspectos que foram considerados pra a realização deste trabalho. Primeiramente abordará-se aspectos do microcontrolador, sua modelagem em software, recursos de *hardware*, para em seguida tratar a interação e a implementação dos outros sistemas.

2.1 MODELAGEM DE SOFTWARE

2.1.1 Modelagem do Microcontrolador

De forma a implementar o carro controlado por voz, foi proposto um diagrama de classes UML, dado pela figura 1. Na figura, temos as classes para gerar a fila de operações realizadas, um *driver* para acionamento de motores, o hospedeiro, podendo este ser aplicado tanto por meio UART ou *Bluetooth*, o receptor de áudio e o modelo interpretador de som baseado em aprendizado de máquina.

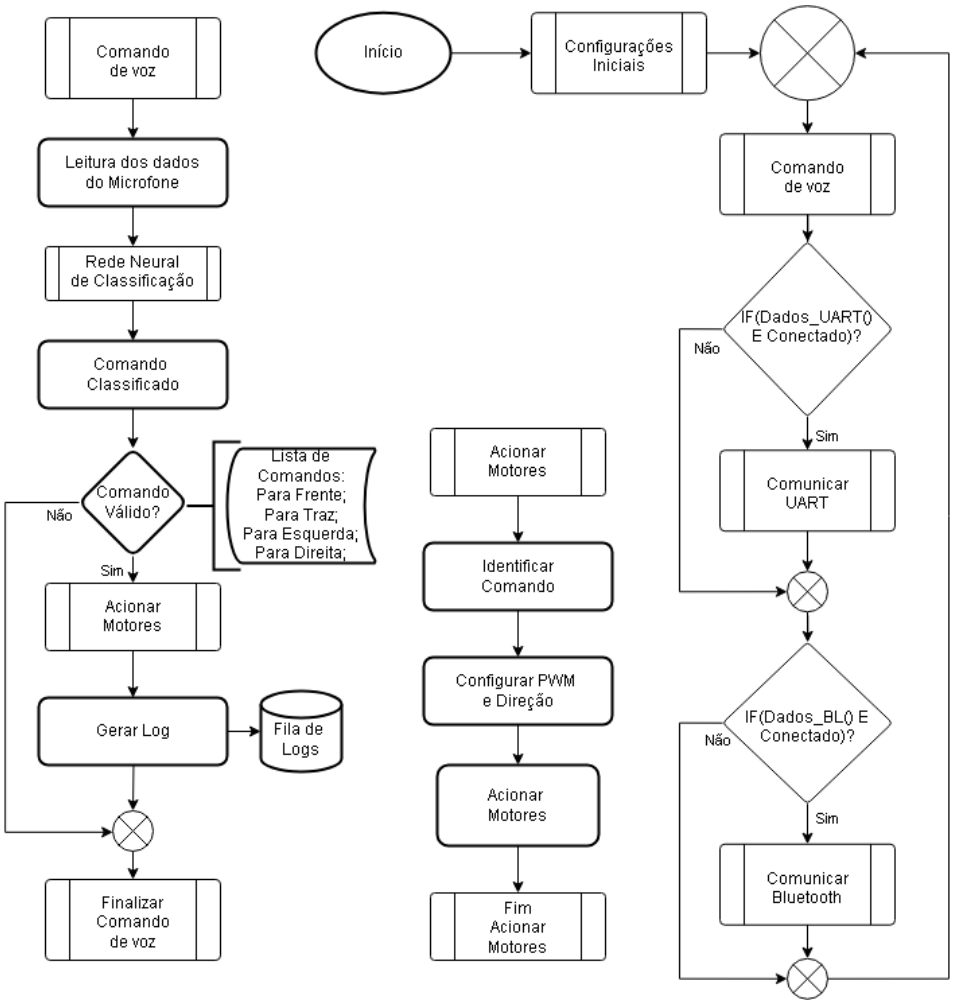
Figura 1 – Diagrama de classes modelando o carro controlado por voz.



Fonte: Os autores

O programa funcionará conforme os fluxogramas encontrados nas figuras 2 e 3. O laço principal se refere a execução padrão do algoritmo, chamando as funções externas sempre que o sistema solicitar.

Figura 2 – Fluxograma representando a sequência de ações do sistema embarcado (Parte 1).



Fonte: Os autores

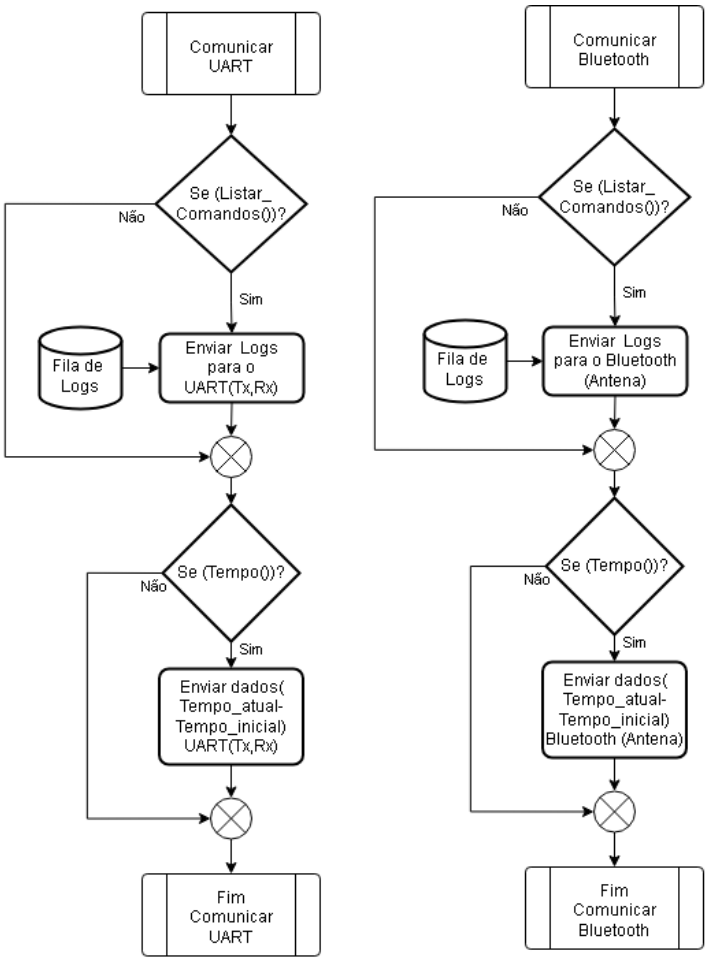
2.1.2 Modelagem do Hospedeiro

A modelagem do sistema hospedeiro, que receberá a lista de operações pela codificação UART ou *Bluetooth*, está representada pelo diagrama UML indicado na figura 4. Tem-se representado no diagrama um hospedeiro que recebe as operações por UART, ou por *Bluetooth*, e imprime os dados recebidos no terminal. A figura 5 mostra um fluxograma para o algoritmo do hospedeiro.

2.2 RECURSOS DE HARDWARE

Para a implementação do mini carro robô controlado por voz, foi optada pela placa da desenvolvimento ESP32 com OLED. A escolha desse microcontrolador envolve os fatores de capacidade de *hardware* tais como memória (520 kB de RAM e até 4 MB Flash) e integração de uma antena para comunicação Bluetooth e Wi-Fi. Além disso, a placa de desenvolvimento possui uma configuração pronta para desenvolver

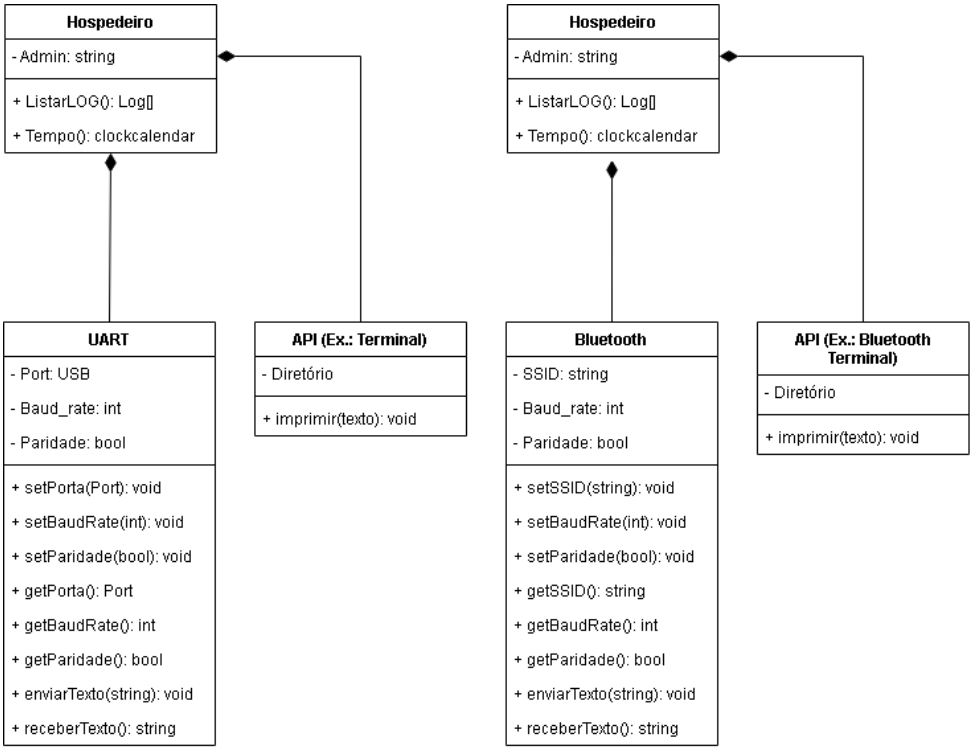
Figura 3 – Fluxograma representando a sequência de ações do sistema embarcado (Parte 2).



Fonte: Os autores

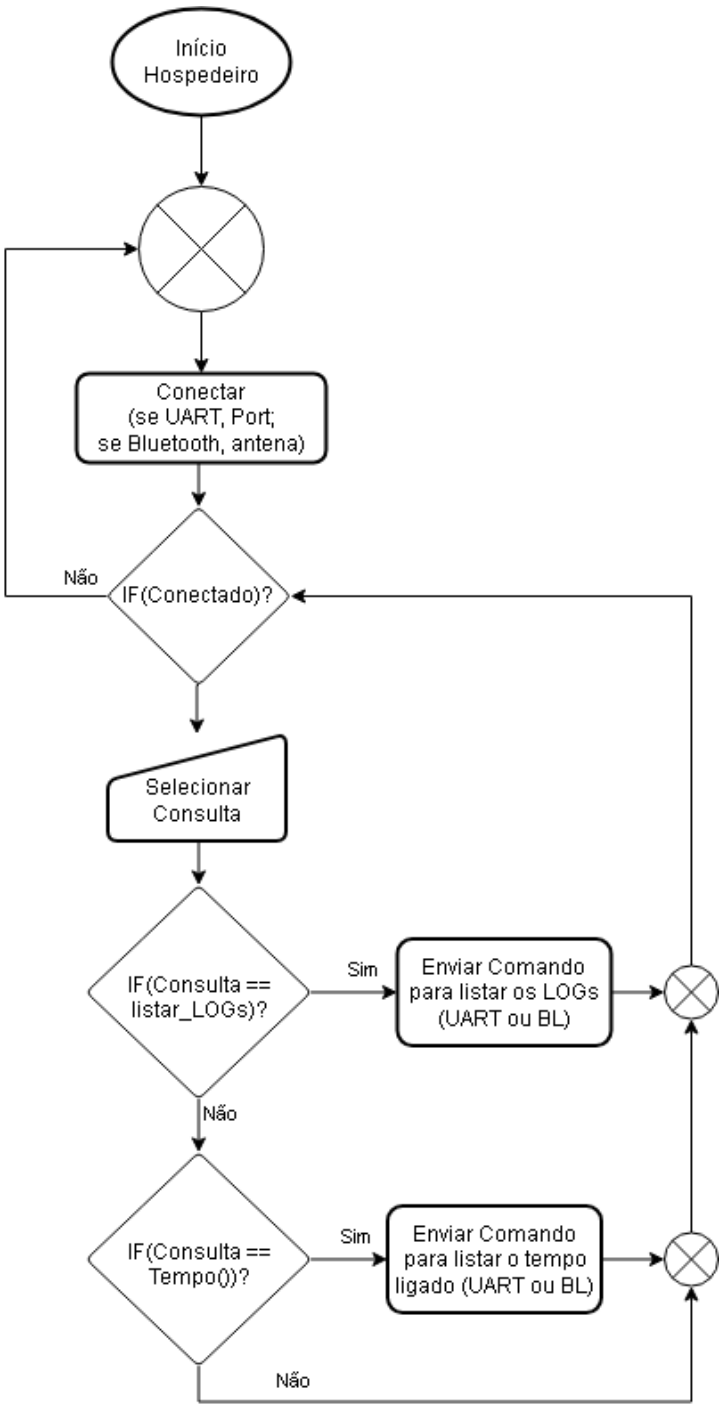
projetos, incluindo um conversor FTDI para facilitar a comunicação através de um cabo USB.

Figura 4 – Diagrama de classes modelando o hospedeiro para carro controlado por voz.



Fonte: Os autores

Figura 5 – Fluxograma do hospedeiro para o projeto do carro controlado por voz.



Fonte: Os autores

3 DESENVOLVIMENTO

Esta seção descreve os procedimentos utilizados para desenvolver o mini-carro robô controlado por voz.

3.1 CÓDIGO DO MICROCONTROLADOR

O código do microcontrolador foi implementado através do Visual Studio Code, com auxílio da extensão platformio. Essa extensão permite obter as especificações de microcontroladores, compilar e enviar o código de forma funcional.

Para a compilação usando um arquivo principal C++, a função principal deve ser declarada com link ao C (extern "C"). Dessa forma, o programa será capaz de utilizar as ferramentas disponibilizadas em C++ pelo projeto.

Seguindo o diagrama de classe proposto pela figura 1, foram montadas as classes para modelar o Motor, Driver dos motores, o microfone, o relógio calendário e a lista encadeada. Uma rede neural convolucional foi treinada e transformada em uma biblioteca para ser adicionada ao projeto com auxílio da ferramenta online Edge Impulse.

O desenvolvimento utilizou como base as diversas bibliotecas e funções disponíveis para abstração e tratamento de registradores de forma abstraída, disponibilizadas pelo fabricante. As bibliotecas auxiliam na configuração e ativação dos periféricos presentes no ESP32, como UART, módulos PWM, etc.

3.1.1 Modelagem do Motor

Para a modelagem do motor, assumiu-se que seus principais atributos que podem ser modelados através do microcontrolador foram, seu ciclo tarefa, a frequência de chaveamento base, a quantidade de pulsos de clock do microcontrolador para um ciclo e a direção de rotação. Esses atributos foram modelados de forma que fosse possível controlar as informações presentes nos registradores do ESP32.

Foram também implementados métodos de redefinição e de obtenção dos parâmetros de forma que fosse possível atualizar eles durante o processamento do código. A classe do motor foi definida como uma classe base virtual de forma que as funções de que executarão o movimento do motor seja definida pelos seus periféricos, ou *drivers*.

3.1.2 Modelagem do Driver

O driver do motor foi modelado como uma classe derivada de motor. Ao herdar os atributos do motor, esta classe serve como uma interface entre o motor e seu periféricos. De forma a trabalhar com a família de microcontroladores ESP, foram

utilizadas as variáveis de configuração e tratamento para do periférico PWM. O ESP32 dispõe de dois módulos de controle de PWM.

No ESP32, cada módulo periférico de PWM dispõe de três temporizadores, os quais geram a frequência de base do sinal modulador PWM, três canais de operadores, que são utilizados para definir as ações a partir do instante em que o contador atinge certo valor (ex.: saída em zero quando o temporizador atingir o valor zero). Cada módulo operador dispõe de duas saídas de sinais, chamadas de geradores, os quais podem ser utilizados de forma a modificar a direção do motor CC, ou controlar os outros tipos de motores.

Esta classe procura abstrair a configuração do PWM para um canal de temporizadores e operadores. A classe permite tanto uma inicialização padrão do PWM, quanto uma inicialização mais detalhada, caso se deseje outros parâmetros de PWM. A classe também conta com métodos para modificar a direção e o ciclo tarefa do periférico.

Este projeto aplicou em suas configurações um clock base com frequência de 1 MHz, uma resolução do contador de 100 pulsos, reduzindo a frequência da onda triangular do PWM para 10 kHz, ciclo tarefa em 50% e gerando formas de onda para enviar o sinal em ambas as direções, para controlar o sentido de rotação do motor.

3.1.3 Modelagem do Microfone

O microfone foi modelado de forma a conter um buffer de sinal em formato de ponto flutuante e a sua frequência de amostragem. Possui uma inicialização padrão utilizando uma frequência de amostragem de 44,1 kHz, valor padrão para diversas aplicações de áudio digital.

Como o microfone funcionará como um intermediário entre o microfone e a rede neural, é importante que ele seja configurado para a mesma frequência de amostragem dada pela qual o classificador de áudio foi configurado. Para este projeto, a frequência de captura para os comandos de voz é de 16 kHz.

O microfone foi modelado de forma abstrata, de forma que a seus valores possam ser facilmente acessíveis através de diversos periféricos encontrados no microcontrolador e, assim, acomodar diversas tecnologias de transmissão de áudio.

3.1.4 Modelagem do Periférico do Microfone

A depender da tecnologia utilizada no microfone, ele pode fornecer informações tanto em formato analógico com em outros formatos de áudio, como o PCM. Diversos microcontroladores dispõem do protocolo de comunicação I2S, uma modificação do protocolo I2C para acomodar sinais de áudio.

Este projeto utilizou o microfone tipo MEMS MSM261S4030H0, que transmite o áudio com codificação PCM24, sendo transmitido primeiramente os bits mais significa-

tivos. O microfone emite saídas nos dois canais de áudio (esquerdo ou direito), sendo cada canal configurado com auxílio da comunicação I2S. O primeiro bit de sinal de um dos canais ocorre no segundo pulso de descida aplicado ao terminal CLK, e após o terminal WS (Word Select) selecionar o canal (baixo para esquerdo e alto para direito).

O ESP32 possui internamente os periféricos necessários para tratar sinais I2S, sendo possível configurá-los para receber diversos tipos de codificação de áudio. Das referências de bibliotecas do ESP32, encontra-se que a codificação desejada para este projeto é a Phillips com um multiplicador de clock de 384. As outras configurações seguem de forma a se ajustar com os parâmetros do microfone.

3.1.5 Rede Neural Classificadora

De forma a obter o reconhecimento de comandos pelo microfone, deve-se primeiramente treinar uma rede neural para depois implementá-la no ESP32. Para tal fim, utilizou-se o *dataset speech commands* (WARDEN, 2018) para treinar a rede neural em conjunto com o Edge Impulse, uma ferramenta online especializada em gerar modelos de rede neural para aplicações embarcadas. Selecionou-se as palavras *up, down, left, rigth, forward, backward e stop* como palavras chave para o modelo.

Foi seguido o procedimento auxiliado pela plataforma online para gerar a rede neural, utilizando um conversor MFCC e uma rede neural convolucional 1D pré configurada com uma camada densa de 128 unidades ao final. Após treinada e validada, foi extraída uma biblioteca C++ da plataforma de forma a incluir ela no projeto.

No código, a biblioteca aguarda a leitura de 16000 amostras do microfone em um segundo para então efetuar a classificação do comando. A rede neural, então, devolve a classe mais provável que seja o som e com o índice de certeza. Ao detectar o comando com uma certeza de 50%, o programa executa uma função correspondente a cada um dos comandos.

3.1.6 Lista encadeada

Para armazenar cada um dos comandos identificados pelo programa, montou-se uma estrutura de dados do tipo lista encadeada. Utilizou-se como base o exemplo dado em aula, com a exceção de que ao invés de um número inteiro, foi modificado para que a lista armazenasse uma estrutura contendo um array de caracteres e um relógio calendário.

Ao se detectar um dos comados com uma certeza satisfatória, o programa acionará os motores de acordo com a função e, automaticamente, executa a chamada para armazenar a estrutura da lista.

3.2 CÓDIGOS

Os códigos até aqui gerados podem ser encontrados a partir do repositório do GitHub localizado no seguinte endereço: https://github.com/renatomeneghin/Mini-Robo-controlado-por-voz.git

REFERÊNCIAS

GALEA, Alex; CAPELO, Luis. **Applied Deep Learning with Python: Use scikit-learn, TensorFlow, and Keras to create intelligent systems and machine learning solutions**. [*S.l.*]: Packt Publishing Ltd, 2018.

LV, Xiaoling; ZHANG, Minglu; LI, Hui. Robot control based on voice command. *In*: 2008 IEEE International Conference on Automation and Logistics. [*S.l.*: *s.n.*], 2008. p. 2490–2494.

SOLOVYEV, Roman; KUSTOV, Alexander; TELPUKHOV, Dmitry; RUKHLOV, Vladimir; KALININ, Alexandr. Fixed-Point Convolutional Neural Network for Real-Time Video Processing in FPGA. *In*: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). [*S.l.*]: IEEE, jan. 2019.

SOLOVYEV, Roman A.; VAKHRUSHEV, Maxim; RADIONOV, Alexander; ROMANOVA, Irina I.; AMERIKANOV, Aleksandr A.; ALIEV, Vladimir; SHVETS, Alexey A. Deep Learning Approaches for Understanding Simple Speech Commands. *In*: 2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO). [*S.l.*: *s.n.*], 2020. p. 688–693.

WARDEN, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. **ArXiv e-prints**, abr. 2018. arXiv: 1804.03209 [cs.CL].