Dispositivo para detecção e classificação de choro de recém-nascidos

São Paulo
2023

Dispositivo para detecção e classificação de choro de recém-nascidos

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista.

São Paulo
2023

Dispositivo para detecção e classificação de choro de recém-nascidos

Versão Monografia Final

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista.

Área de Concentração:
Inteligência artificial embarcada

Orientador:
Bruno de Carvalho Albertini

Co-orientador:
Roseli de Deus Lopes

São Paulo
2023

RESUMO

Este projeto visa a criação de um dispositivo embarcado capaz de identificar a causa do choro de recém-nascidos, classificando-os em classes como fome, dor, surdez, entre outros, por meio de algoritmos de machine learning embarcados. Estudos existentes mostram que essa classificação é possível, e modelos de algoritmos foram criados para tal tarefa. Entretanto, tais modelos são computacionalmente custosos de tal forma a ser inviável a implementação em um dispositivo embarcado com limitações de processamento e memória. Portanto, é escopo deste trabalho o estudo e a elaboração de um hardware com capacidade suficiente para esse processamento de forma a aproximar o resultado de um produto comercialmente viável, juntamente com a implementação dos algoritmos de machine learning para tal classificação com precisão comparável aos modelos existentes, assim como uma validação dos resultados obtidos.

Palavras-Chave – Machine Learning, Embarcados, Recém-Nascido, Choro, Sensoriamento.

SUMÁRIO

1 Introdução **6**

1.1 Motivação 6

1.2 Objetivo 7

1.3 Justificativa 7

1.4 Organização do Trabalho 8

2 Aspectos Conceituais **9**

2.1 Aquisição de sinais analógicos 9

2.2 Pré processamento 9

2.3 Espectrograma 10

2.4 RNN e LSTM 11

3 Metodologia do Trabalho **12**

4 Especificações e Requisitos **13**

4.1 Requisitos de Produto 13

4.2 Requisitos de Hardware 14

4.2.1 Visão Geral 14

4.2.2 Alimentação 15

4.2.3 Microcontrolador 15

4.2.4 Microfone 16

4.2.5 LEDs 16

4.2.6 ESP32-CAM 17

5 Desenvolvimento do Trabalho **18**

5.1 Tecnologias Utilizadas 18
5.1.1 Microcontrolador 18
5.1.2 Microfones 19
5.1.3 Circuito de Alimentação 19
5.1.4 Keras 19
5.1.5 CMSIS DSP 19
5.1.6 STM32CubeMX e X-CUBE-AI 20
5.2 Projeto de Hardware 20
5.2.1 Circuito Saturador 20
5.2.2 Circuito Amplificador 21
5.2.3 Circuito de Alimentação 23
5.2.3.1 Escolha da Bateria 23
5.2.3.2 Circuito da Bateria e da Alientação Externa 25
5.2.3.3 Regulador 3.3 Volts 26
5.2.4 Circuito do Microcontrolador 27
5.2.4.1 Alimentação do Microcontrolador 29
5.2.4.2 Escolha de pinos 31
5.2.4.3 Interface de Programação 31
5.2.5 Design da PCB 32
5.3 Projeto de Software 35
5.3.1 Pré processamento 35
5.3.2 Modelo de classificação 36
5.3.3 Dados de treino 38
5.3.4 Data Augmentation 39
5.4 Projeto de Firmware 40
5.4.1 Arquitetura 40

5.5 Projeto Mecânico 42
5.6 Implementação do Hardware 43
5.6.1 Pedido de placas e componentes 43
5.6.2 Soldagem das placas 43
5.6.3 Bring Up e Testes dos Módulos 46
5.6.3.1 LEDs 46
5.6.3.2 Circuito Saturador 47
5.6.3.3 Circuito Amplificador 47
5.7 Implementação de Software 48
5.7.1 Treinamento 48
5.8 Implementação do Firmware 49
5.8.1 Aquisição de som 49
5.8.2 Pré-processamento 51
5.8.3 Rede Neural embarcada 51
5.9 Implementação da Peça Mecânica 52

6 Considerações Finais **53**
6.1 Conclusões 53
6.2 Contribuições 53
6.3 Perspectivas de Continuidade 53

Apêndice A – Pesquisa de Mercado **55**
A.1 Gráficos 55

Referências **57**

1 INTRODUÇÃO

1.1 Motivação

Cuidar de recém-nascidos não é uma tarefa trivial. Compreender as suas necessidades a partir de seus choros costuma ser uma problemática frequente, na qual o ser humano no geral não é muito eficiente. Estudos demonstram que pessoas treinadas com esse intuito atingem uma acurácia de 30,33% (MUKHOPADHYAY et al., 2013). Além disso, essa classificação exige a constante vigilância do agente, que normalmente acaba se acostumando com o choro do recém-nascido que tem mais contato. Uma solução utilizando machine learning vem de encontro com esses desafios, de forma que pode ser feita por um dispositivo de forma automatizada e constantemente.

Além disso, o sistema teria benefícios do processamento totalmente embarcado, como uma maior confiabilidade e disponibilidade, pela não necessidade de conectividade com internet, assim como uma garantia de maior segurança e privacidade.

Estudos recentes conseguiram atingir índices de precisão que variam de 71.68% a 94.97% (JI et al., 2021)(LAVNER et al., 2016). Tendo esses estudos como referências, vê-se a possibilidade de embarcar uma solução de forma a criar um dispositivo capaz de realizar o processo de classificação dos choros. Porém, é constatado que tais redes podem ser computacionalmente custosas, o que gera a necessidade de um processo de otimização para que o projeto seja viável e consiga operar dentro dos limites dos hardwares disponíveis. Observa-se uma tendência na literatura de uso de CNNs e RNNs como arquiteturas utilizadas para a abordagem do problema, poucas camadas dessas redes já podem levar a grandes requisitos de memória e processamento, chegando na ordem de 50 milhões de parâmetros treináveis (FRANTI et al., 2018), que equivale a 190,7 MB de memória necessária para armazenar o modelo, salvo otimizações, valor fora do comum entre os dispositivos embarcados comerciais.

1.2 Objetivo

É, portanto, escopo deste projeto a criação de um dispositivo capaz de indicar a causa do choro de um recém nascido e que seja comercialmente viável, sendo capaz de concorrer com produtos de cuidados de bebês tanto em questão de preço quanto de funcionalidades e inovações.

Visto que esse produto se posicionaria no mercado de forma muito próxima a uma babá eletrônica, também faz parte do projeto a implementação de funcionalidades presentes nesses produtos que já estão presentes no mercado, e acrescentar à elas a tecnologia da classificação do choro. O público alvo desses produtos são pais e mães que utilizariam o produto em suas casas.

A abordagem escolhida para solucionar a grande gama de problemas é a de desenvolver uma linha de produtos hierárquica, onde produtos mais avançados contém mais características de babás eletrônicas, e mais básicos contem apenas o modelo de classificação.

1.3 Justificativa

Uma correta classificação do desconforto do recém nascido pode gerar uma melhora na qualidade de vida, na medida em que os pais podem tratar com maior urgência casos que julgarem mais graves. A um bebê com suspeitas de doenças deve, deve-se estar muito mais atento a choros relacionados a dor, e é de grande ajuda para esse cuidado um classificador totalmente automatizado constantemente ouvindo.

Ao realizarmos uma breve pesquisa de mercado, constatamos que 91,8% dos entrevistados já tiveram problemas na hora de diagnosticar o motivo do bebê que estavam cuidando estar chorando, e 77,6% considera que ter um dispositivo o qual ajudasse a classificar os choros do recém-nascido ajudaria **muito** no dia-a-dia. Ainda nessa mesma pesquisa, observamos que 42,9% dos entrevistados que já cuidaram de bebês nunca utilizaram babás eletrônicas, e que entre os motivos estão a **falta de acessibilidade** e o **custo elevado** dos produtos existentes no mercado (Apêndice A).

Juntando as informações coletadas mais a carência de novas tecnologias no mercado de produtos para cuidados de bebês, observa-se uma grande oportunidade no mercado ao oferecermos produtos nessa categoria em diversas faixas de preço, se tornando acessível a maioria dos consumidores, e ainda fornecendo uma grande inovação tecnológica,

justificando assim a escolha do projeto.

1.4 Organização do Trabalho

Deve-se então, criar um novo modelo de classificação com menores requisitos de hardware, mantendo uma precisão compatível com os modelos já existentes. Serão utilizados como base as diversas abordagens já utilizadas na literatura, mantendo como objetivo principal a economia de recursos computacionais.

Também está entre os trabalhos o projeto de um hardware com capacidade suficiente para o processamento. Além dos requisitos exigidos pela rede neural, o dispositivo deve ser capaz de monitorar constantemente o recém-nascido. Dessa maneira, esse deve coletar sinais sonoros do ambiente constantemente. Ao embarcar a solução, diversos requisitos não funcionais do sistema se torna relevante, como consumo de energia e tempo de inferência. Além disso, é necessário desenvolver métodos de sinalização do choro classificado, para o usuário conseguir visualizar facilmente.

Juntamente, as demais funções de babás eletrônicas devem ser implementadas, tanto em software como em hardware.

2 ASPECTOS CONCEITUAIS

2.1 Aquisição de sinais analógicos

Para aquisitar os sinais sonoros dos choros dos bebês é preciso converter o sinal do microfone, analógico, para um sinal digital, o qual é possível ser processado pelo microcontrolador. Para isso, o microcontrolador possui um periférico chamado ADC (Analog Digital Converter) que faz a conversão do sinal analógico para um sinal digital de número de bits definido, sendo que quanto maior o número de bits, maior a resolução do sinal adquirido.

Junto a isso, é necessário adquirir o sinal a uma taxa de amostragem alta e constante para que não ocorra distorções no sinal. Porém, o processo de salvar a aquisição na memória do microprocessador pode ser lento demais e acabar prejudicando o processamento do sinal. Para resolver esse problema, o microcontrolador possui outro periférico chamado DMA (Direct Memory Access) o qual possibilita que as informações aquisitadas pelo ADC sejam escritas diretamente na memória, sem precisar usar o processamento do processador. Dessa forma, é possível chegar a uma taxa de amostragem extremamente superior se utilizamos os métodos convencionais de acesso à memória.

2.2 Pré processamento

O áudio é codificado digitalmente por uma sequencia numérica cujo comprimento é função da taxa e do tempo de aquisição. Tomando como exemplo o formato de arquivo *.wav*, tem-se como frequência de aquisição padrão 44,1 kHz. Desse modo, a cada segundo de gravação de áudio a sequência aumenta seu tamanho em 44100 amostras.

Como o objetivo do projeto é utilizar os dados de áudio como entrada em um modelo de machine learning, se torna necessária a redução da quantidade de valores numéricos, para viabilização da rede neural utilizada. Dito isso, a etapa de pré processamento deve fazer a extração de atributos, codificando o dado de áudio em menos dados numéricos que

carreguem o significado a ser analisado pela rede neural.

2.3 Espectrograma

Uma forma de extração de atributos utilizada é o espectrograma. Esse é um diagrama que analisa, em diferentes janelas de tempo, o sinal no domínio da frequência.

Com isso, dado um sinal de áudio de período t segundos sendo amostrado em uma taxa de f Hz. A operação de criação do espectrograma realiza a seguinte transformação.

$$\mathbb{R}^t \to \mathbb{M}_{t/T \times f/2}(\mathbb{R})$$

Sendo T o período das divisões realizadas pela operação.

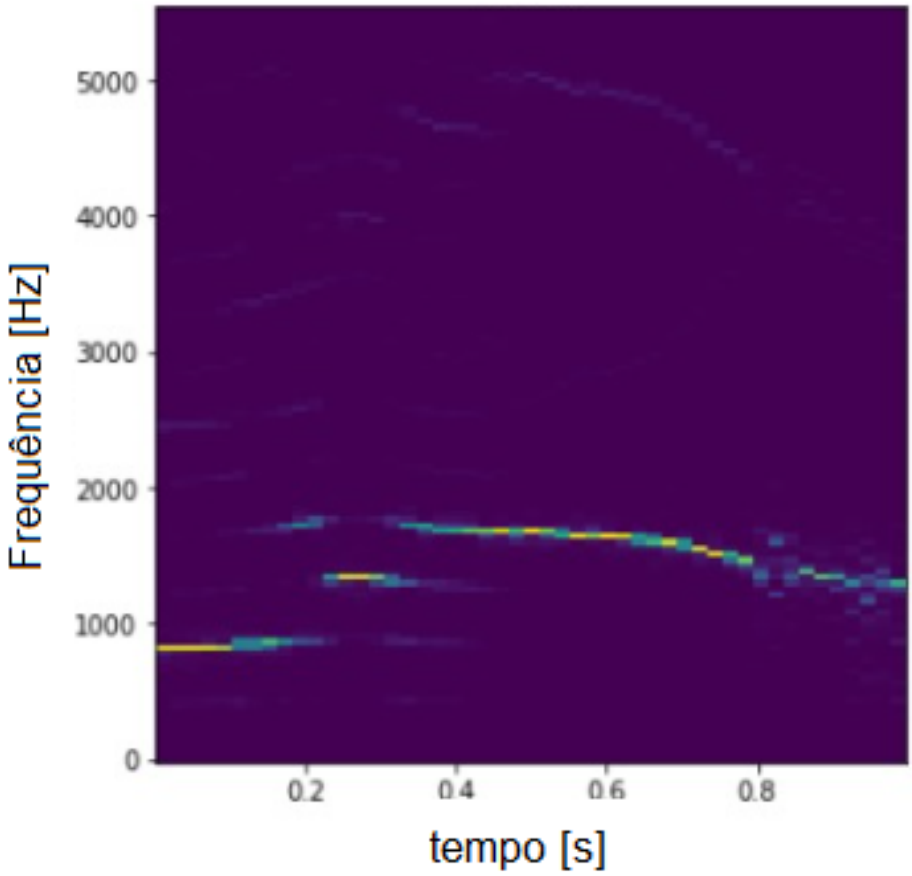

Figura 1: Exemplo de espectrograma, utilizando áudio de choro contido nos dados de treino.

Como observado na **Figura 1**, através do espectrograma é possível identificar visualmente o comportamento do áudio, tal que frequências mais altas representam sons mais

agudos, e mais baixas mais graves, e é possível ver a transição entre essas frequências ao longo do tempo de forma bem mais direta que analisando o sinal puramente em função do tempo.

Para fins de simplificação, é possível também limitar superiormente as frequências a serem analisadas, sem perdas relevantes de informação se as amostras não apresentarem frequências elevadas. Nesse caso, obtemos uma simplificação dos dados de entrada reduzindo seu tamanho.

2.4 RNN e LSTM

Modelos de redes neurais aplicados a processamento de linguagem natural, texto, e áudio devem resolver problemas com requisitos particulares. Entre eles, a necessidade de tratar dados de entrada de tamanho variável e reconhecer padrões sequenciais em dados. Desse modo, é comum na literatura o uso de arquitetura de redes neurais RNN (*Recurrent Neural Network*) como abordagem para tais classes de problemas (RUMELHART; MCCLELLAND, 1987).

Em particular, uma modificação da arquitetura RNN é a LSTM (*Long Short-term Memory*). Que acrescenta à RNN mecanismos para memorização de informações de longo prazo, permitindo que dados distantes interfiram entre si, ao longo da propagação da rede, facilitando assim o reconhecimento de padrões (HOCHREITER; SCHMIDHUBER, 1997).

3 METODOLOGIA DO TRABALHO

Os diversos estudos sobre o problema de classificação e/ou identificação de choros de recém nascidos mostram a viabilidade de execução deste projeto e desse modo, esses estudos indicam modelos de classificadores que serão utilizados como base para a elaboração do modelo (MUKHOPADHYAY et al., 2013).

Esses estudos também indicam que a falta de dados para o treino de modelos de machine learning é um problema recorrente nessa área(RUSU et al., 2015), então, devemos buscar acesso às *databases* utilizadas nos estudos citados. Dentre os dados mais utilizados, existem o *BabyChillanto* (REYES-GALAVIZ et al., 2004) obtido pelo "National Institute of Astrophysics and Optical Electronics, CONACYT Mexico", e o *SPLANN* obtido por “Sf. Pantelimon Emergency Clinical"o (BğNICğ et al., 2016) . E indo além dos datasets, para termos uma maior variedade de inputs para o treinamento de rede, é possível utilizar métodos de *Data Augmentation* (KACHHI et al., 2022) para dar um reforço no processo de treinar a rede.

Em paralelo, devemos listar os requisitos funcionais e não funcionais do dispositivo. Em seguida, deve-se fazer o projeto de hardware buscando atender a tais requisitos.

Em seguida testes serão desenvolvidos em uma plataforma de desenvolvimento, de forma que não tenhamos muita limitação de processamento inicial, nem dificuldades eletrônicas para a obtenção do som.

Após isso, começará o desenvolvimento do hardware dedicado, juntamente com os trabalhos de integração.

4 ESPECIFICAÇÕES E REQUISITOS

4.1 Requisitos de Produto

Será desenvolvido uma linha de produtos, que contém três modelos. A definição dos requisitos dos produtos foi pensada de forma a visar uma modularidade entre esses produtos. Dessa forma, a passagem de um produto para o outro deve exigir um pequeno esforço, como apenas conectar ou desconectar módulos, agilizando assim uma possível linha de produção por meio de padronização dos produtos.

Para garantir a modularidade, os produtos terão cada vez mais recursos ao longo da linha de produtos, mantendo os recursos dos produtos anteriores. Foi elaborado então os seguintes modelos:

Nome	**Recursos**
CryCare Compact	Reconhecimento e classificação de choro de recém nascidos
CryCare B-Mo light	Conectividade
CryCare B-Mo Plus	Transmissão de imagens

Tabela 1: Definição de recursos para linha de produtos.

Tendo então o escopo geral dos produtos, podem ser definidos os requisitos funcionais de cada modelo:

Modelo	Requisitos funcionais	Requisitos não funcionais
CryCare Compact	- Identificação e classificação de choros - Indicação visual por meio de LEDs - Funcionamento utilizando Bateria - LEDs de status	- Tempo de resposta < 2 s - Duração de bateria > 48 H
CryCare B-Mo Ligth	- Aplicativo para visualização e controle - Conectividade via rede WIFI - Comunicação bidirecional de áudio - Possibilidade de tocar musicas - Sensor de temperatura ambiente	
CryCare B-Mo Plus	- Câmera de monitoramento - Sensor de câmera noturna - Visualização de vídeo por aplicativo	

Tabela 2: Definição de requisitos para linha de produtos.

4.2 Requisitos de Hardware

O Hardware será responsável por fazer a aquisição do som, realizar o processamento e a inferência, assim como dar o feedback visual para o usuário. Além disso, deverá ter uma interface para se comunicar com uma ESP32-CAM que será responsável pelos recursos presentes nos produtos mais caros.

4.2.1 Visão Geral

O projeto foi então dividido nos módulos de: Alimentação, Microcontrolador, Microfone, LEDs e ESP32-CAM.

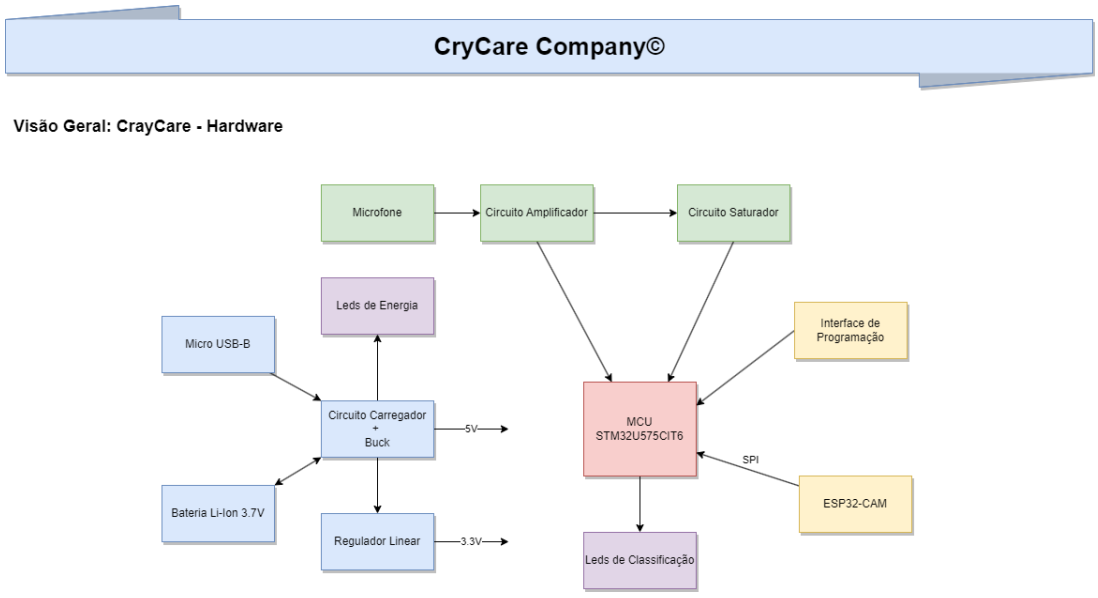

Figura 2: Diagrama da arquitetura geral do hardware a ser desenvolvido

4.2.2 Alimentação

Conforme descrito na **Tabela 2**, o dispositivo deve ser capaz de ser alimentado por uma bateria, que deve ser recarregada via USB. Dessa forma, foram estabelecidos os requisitos:

- A placa deverá suporta uma bateria recarregável para fornecer energia ao sistema
- A placa deverá suportar ser alimentada via USB
- A placa deverá fazer o carregamento da bateria via USB
- A placa deverá Identificar automaticamente quando está sendo alimentada pelo USB ou pela bateria
- A placa deverá regular a tensão da bateria para 5V
- A placa deverá regular os 5V para 3.3V para alimentar o MCU
- A placa poderá ser desligada completamente a partir de um Switch

4.2.3 Microcontrolador

Grande parte da memória flash será utilizada para armazenar a rede neural de classificação. Então, foi feito um modelo de RNN inicial, e compilado utilizando ARM-GCC para descobrir o consumo de memória do programa.

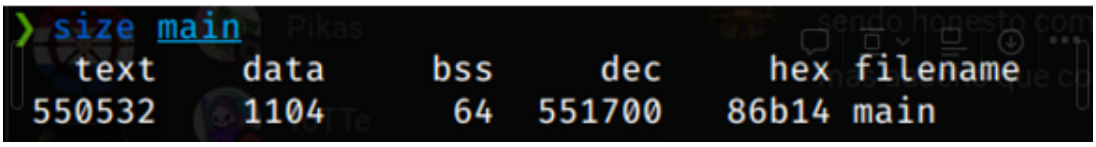

Figura 3: Dados de binário compilado com rede neural de classificação inicial

Observou-se então que o programa necessita de aproximadamente 500 kB de memória flash. Visto que a rede testada era simples e não atingiu a precisão esperada, o requisito definido foi de 4 vezes maior, 2 MB.

Além disso, será necessário armazenar os dados coletados do microfone. Supondo que a rede de classificação utilizará 1 segundo de amostragem de sinal, que cada sinal é um inteiro de 2 bytes, e que o sinal será amostrado em 44 kHz:

$$n = 2 \cdot f \cdot \Delta t = 2 \cdot 44000 \cdot 1 = 88000\,B$$

$$n = 86\,kB$$

Como margem de segurança, foi considerado o dobro deste valor para amostragem de sinal: 172 kB. Para a execução geral do programa então, foi decidido o requisito de 250 kB.

- A placa deverá usar um microcontrolador com pelo menos 2 MB de memória Flash, 250 KB de memória RAM e suportar modos Low Power
- A interface de programação do microcontrolador deverá ser a SWD (serial wire debug)

4.2.4 Microfone

Para cumprir o requisito não funcional de duração de bateria, será necessário utilizar o modo de baixa energia do processador. Para isso, é necessário um sinal do microfone, para acordar o processador. Por isso, será utilizado juntamente com o microfone, um circuito saturador, responsável por gerar um sinal lógico positivo caso o microfone detecte um som alto o suficiente.

- A placa deverá capturar o sinal do microfone e amplificá-lo para a leitura no micro
- A placa deverá criar um sinal saturado com a entrado do microfone para a função de wake-up

4.2.5 LEDs

- A placa deverá indicar o estado da bateria através de LEDs brancos
- A placas deverá expor os resultados da classificação a partir de LEDs de diferentes cores
- A placa deverá possuir um LED para indicar o estado de funcionamento do produto (ON/OFF)

4.2.6 ESP32-CAM

A aquisição de imagens será feita utilizando ESP32-CAM. Além disso, a ESP32 será responsável por fazer a medida de temperatura e a comunicação WIFI. Dessa forma, como mostra a **Figura 4**, é possível atingir uma grande modularidade entre os 3 modelos de produto, visto que as funcionalidades presentes no *CryCare B-Mo Light* são alcançáveis apenas adicionando a ESP32-CAM, e as do *CryCare B-Mo Pro* apenas adicionando a camera à ESP32.

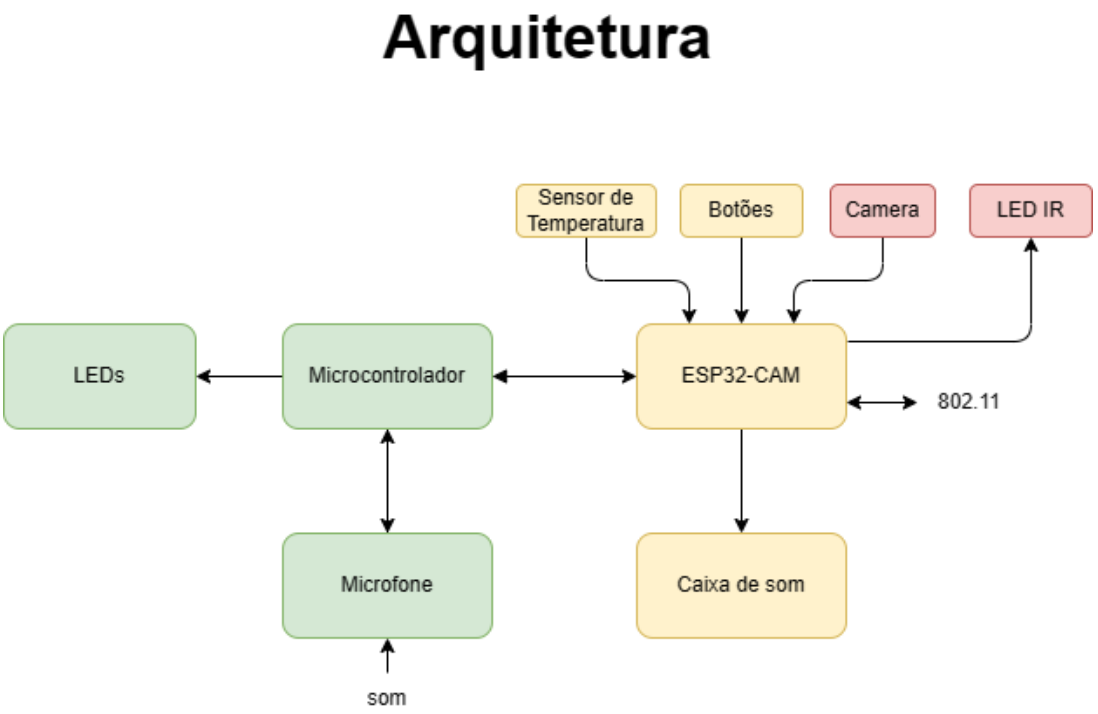

Figura 4: Arquitetura de hardware

- A placa deverá fazer interface com uma ESP32-CAM

5 DESENVOLVIMENTO DO TRABALHO

5.1 Tecnologias Utilizadas

5.1.1 Microcontrolador

Para suprir os requisitos de memória e processamento, optamos por utilizar o microcontrolador da empresa STMicroeletronics STM32U575CIT6. Os motivos que embasaram a escolha desse microcontrolador em especifico foram:

1. Memória Flash: O micro em questão possui um total de 2 Mbytes de memória flash, quantidade suficiente para armazenar a rede neural que será utilizada e todos seus pesos.

2. Memória RAM: O micro em questão possui um total de 786 Kbytes de memória RAM, quantidade suficiente para armazenar os dados de pré e pós processamento utilizados durante a inferência da rede neural.

3. Processamento: O micro em questão possui um clock que pode chegar até 160 MHz, quantidade suficiente para realizar a inferência da rede neural em um tempo desejado no produto. Além disso, o micro conta com uma arquitetura Cortex-M33 que possui instruções específicas e otimizadas para processamento digital de sinal (DSP) que beneficiará os pré-processamentos que serão necessários fazer durante a aquisição dos sinais de som.

4. Consumo: O micro em questão possui um modo de operação Ultra Low Power, que garantirá que a bateria do produto durará durante muito tempo enquanto ele detectar nenhum som de choro.

5. Suporte e Familiaridade: A ST possui uma vasta quantidade de soluções que facilitam o desenvolvimento do firmware desejado, como por exemplo, um ambiente em que podemos adaptar uma rede neural feita em Keras (Python) para código de má-

quina (Assembly) utilizados pelo microcontrolador. Além também de já estarmos familiarizados com o ambiente de desenvolvimento dos microcontroladores da ST.

5.1.2 Microfones

Para realizar a captura dos sinais sonoros iremos utilizar 2 microfones de eletreto, e 2 circuitos principais, um circuito saturador **??** para fornecemos um nível lógico alto para despertar o microcontrolador quando for detectado um som elevado (podendo ser um choro), e um circuito amplificador **??** que servirá para fornecer um ganho no sinal do microfone para que seja possível o microcontrolador processar o sinal sonoro.

5.1.3 Circuito de Alimentação

Para o circuito de alimentação utilizaremos um CI presente em dispositivos Power-Bank **??** que realizam o gerenciamento e o carregamento de baterias. Além disso, utilizaremos um regulador linear para transformar os 5V da entrada do circuito carregador mais entrada USB em 3,3V que alimentarão todo o resto do circuito.

5.1.4 Keras

O desenvolvimento e treinamento da rede neural foi realizado utilizando Keras, uma API de deep learning na linguagem Python, desenvolvida sobre a biblioteca TensorFlow. Foi escolhida devido à sua simplicidade e eficiência, possibilitando um rápido desenvolvimento do modelo.

5.1.5 CMSIS DSP

Buscando eficiência no processamento embarcado, foi utilizado bibliotecas da CMSIS ("Cortex Microcontroller Software Interface Standard"), padrão criado pela ARM que define camadas de abstração de hardware para processadores da linha Cortex-M, dessa forma, as bibliotecas são capazes de utilizar instruções específicas desta arquitetura, aprimorando a eficiência do código gerado.

Em particular, é utilizada a biblioteca de processamento de sinais digitais, CMSIS DSP. Nela, são implementada genericamente funções como filtros, transformadas e operações em matrizes e vetores. As funções de FFT ("Fast Fourier Transform") e operações

matriciais foram amplamente utilizadas no projeto para a realização do pré-processamento dos sinais de áudio.

5.1.6 STM32CubeMX e X-CUBE-AI

O software STM32CubeMX, da empresa STMicroeletronics foi utilizada durante o desenvolvimento. Essa ferramenta fornece geração automática de código para camada de abstração de hardware, fornecendo assim códigos de baixo nível para comunicação com os periféricos utilizados, como portas de entrada e saída e conversores analógico-digitais.

Adicionalmente, o pacote de expansão X-CUBE-AI possibilitou a conversão do modelo de rede neural já treinado produzido pelo Keras, para um conjunto de bibliotecas pré compiladas e cabeçalhos na linguagem C.

5.2 Projeto de Hardware

As tecnologias citadas foram incorporadas em uma placa de circuito impresso que será responsável por todas as funcionalidades do projeto em questão.

O projeto da placa foi feito utilizando o Software Altium Designer, onde foi possível produzir tanto o esquemático, quanto o design da placa de circuito impresso.

Como a placa possui diversos módulos, dividimos o circuito em blocos menores.

5.2.1 Circuito Saturador

O circuito saturador foi baseado no módulo KY-037, um conhecido sensor de som utilizado em diversas aplicações.

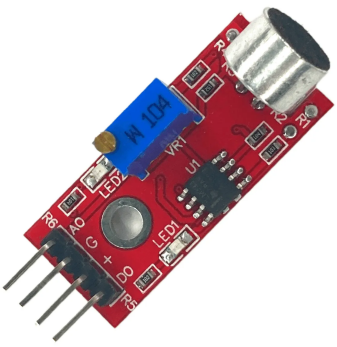

Figura 5: Módulo Sensor de Som: KY-037

Trata-se de um circuito comparador utilizando um amplificador operacional, onde na entrada positiva alimentaremos com 1/2 Vcc (1.65 Volts) e na entrada negativa estará a saída do microfone polarizado pelo resistor R7. A sensibilidade do circuito se baseia no valor desse resistor. Para definir o valor do resistor, utilizamos o próprio módulo KY-037 que possui um potenciômetro, então o ajustamos até a sensibilidade desejada e medimos o valor da resistência obtida. Dessa forma, chegamos a um valor de 6k Ohms.

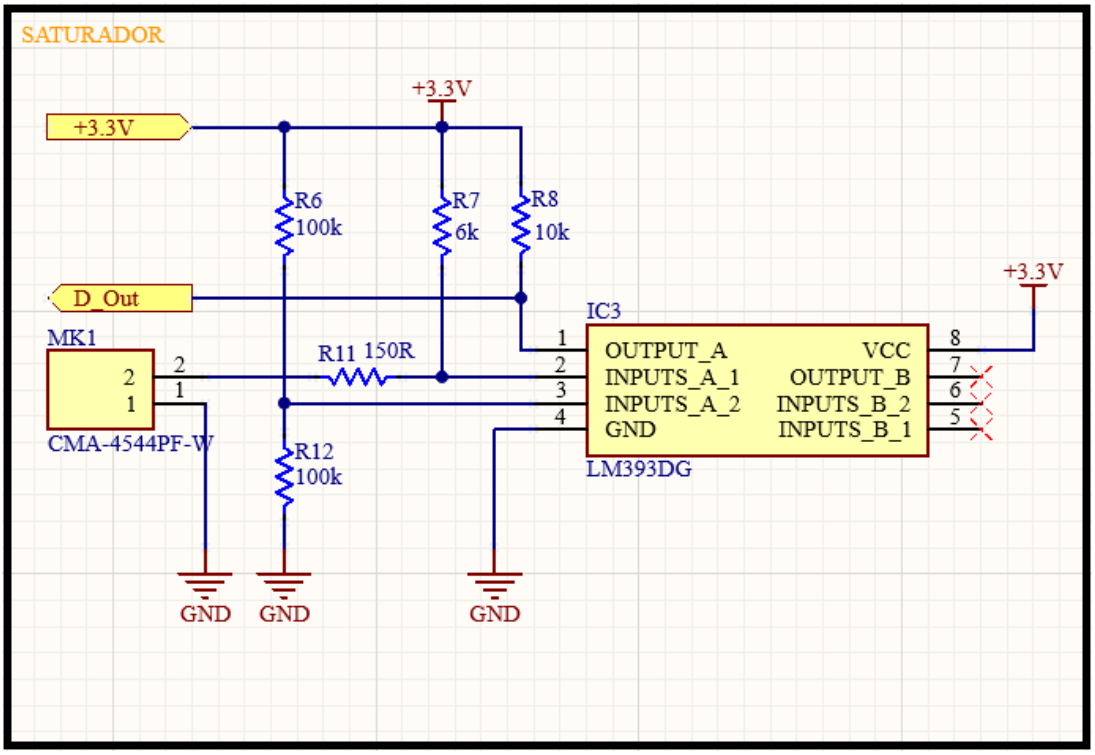

Figura 6: Esquemático do Circuito Saturador

Dessa forma, o circuito ficará sempre com a saída em nível alto, e quando o microfone polarizado fornecer uma tensão maior que 1.65 Volts a saída do amplificador operacional cairá para nível baixo gerando uma interrupção no microcontrolador.

5.2.2 Circuito Amplificador

O circuito amplificador foi baseado em um módulo de microfone Adafruit-MAX9814. Este módulo conta com o CI MAX9814ETD+T um amplificador mono canal integrado com até 60 dB de ganho.

A implementação foi feita seguindo a Typical Application presente no datasheet do componente com algumas pequenas alterações.

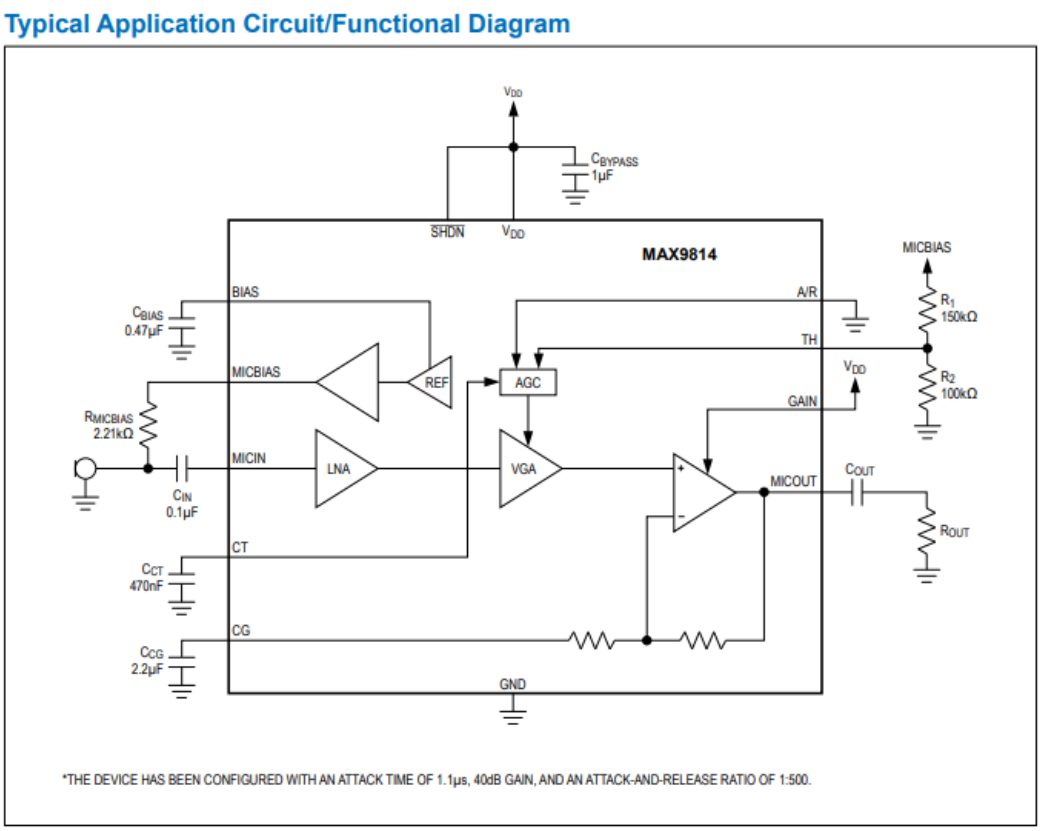

Figura 7: Typical Application Circuit/Functional Diagram

Para nossa aplicação, optamos por deixar os pinos Gain e A/R flutuando para obter um ganho de 60dB e uma proporção entre attack e release de 1:4000, conforme informado no datasheet.

9	A/R	Trilevel Attack and Release Ratio Select. Controls the ratio of attack time to release time for the AGC circuit. A/R = GND: Attack/Release Ratio is 1:500 A/R = V_{DD}: Attack/Release Ratio is 1:2000 A/R = Unconnected: Attack/Release Ratio is 1:4000
10	GAIN	Trilevel Amplifier Gain Control. GAIN = V_{DD}, gain set to 40dB. GAIN = GND, gain set to 50dB. GAIN = Unconnected, uncompressed gain set to 60dB.

Figura 8: Informações sobre configuração do MAX9814

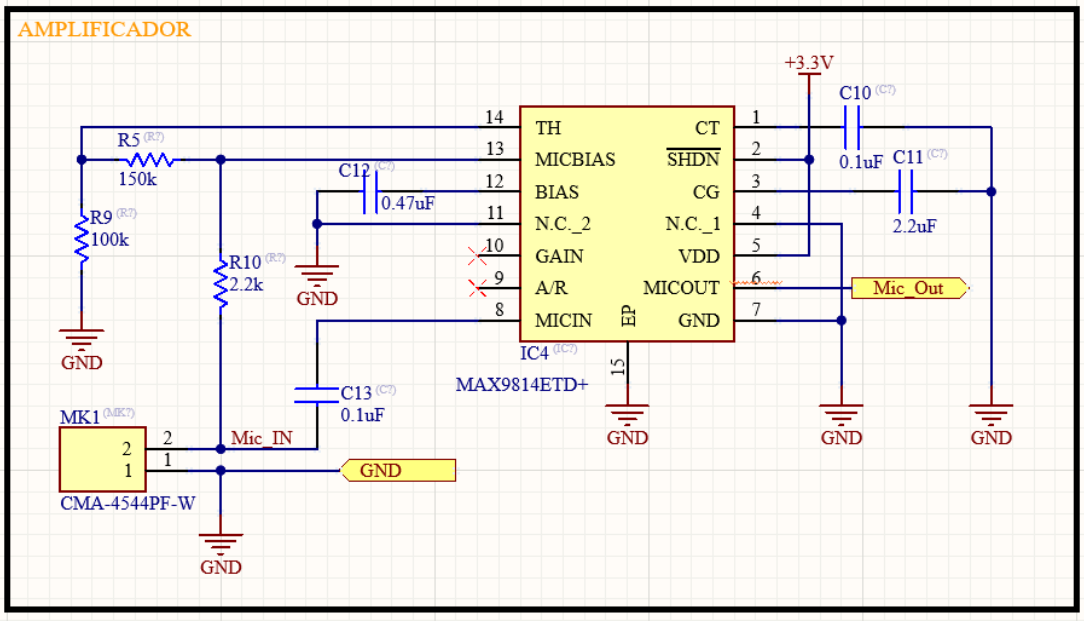

Figura 9: Esquemático do Circuito Amplificador

5.2.3 Circuito de Alimentação

O Circuito de alimentação foi pensado para que a placa funcione tanto na tomada quanto por bateria. Para isso, utilizamos o CI IP5306 4.35V, que é responsável por fazer o gerenciamento entre a alimentação externa e a alimentação da bateria. Este também faz o gerenciamento de carga e descarga da bateria de Lítio responsável pelo funcionamento da placa fora da tomada. Esse circuito principal fornece uma saída de 5 Volts, porém precisamos de 3.3V para que o resto da placa funcione corretamente, portanto utilizamos um regulador linear para converter a tensão para o resto do circuito.

5.2.3.1 Escolha da Bateria

Para definir a duração e a capacidade da bateria é necessário levar em conta o consumo de corrente do circuito. Para as contas que faremos a baixo será levado em conta o consumo máximo de todo o circuito.

Modo Runing

O Modo Running se refere a quando micro está processando o audio efetivamente e mostrando o resultado em tempo real.

Microcontrolador

De acordo com o datasheet do micro STM32U575CI6T o consumo de corrente em

Run Mode é de 19.5uA por MHz de clock. Sabendo que o clock mais rápido do micro é de 160MHz temos:

$$C = 19.5\mu A/MHz * 160Hz\ C = 3.12mA$$

Circuito Amplificador do Microfone

De acordo com o datasheet do MAX9813 o consumo máximo do CI é de 6mA.

Circuito Saturador do Microfone

De acordo com o datasheet do LM393 o consumo máximo do CI é de 1mA.

LEDs

Levando em conta que apenas 2 LED ficarão acesos no modo de operação Running (Indicação + ON/OFF), temos 20mA

Modo Sleep

O Modo Sleep se refere a quando o micro está hibernando esperando ser acordado pelo sensor de som.

Microcontrolador

De acordo com o datasheet do micro STM32U575CIT6 o consumo de corrente no modo Standby é de 210nA (com 24 wake-up pins)

Circuito Amplificador do Microfone

De acordo com o datasheet do MAX9813 o consumo máximo do CI é de 6mA.

Circuito Saturador do Microfone

De acordo com o datasheet do LM393 o consumo máximo do CI é de 1mA.

LEDs

Levando em conta que apenas 1 LED ficará acesos no modo de operação Sleep (ON/OFF), temos 4mA

Carga da Bateria

Levando em conta que queremos que a bateria dure 2 dias, a equação para calcular a carga se da:

$$Carga = Consumo * Tempo$$

Já que esperasse que o produto passe metade do tempo funcionando e metade em modo sleep, temos a carga total:

$$CargaTotal = (11mA * 24h) + (30.12mA * 24h) = 988.8mAh$$

5.2.3.2 Circuito da Bateria e da Alientação Externa

Para esse módulo, utilizamos o CI IP5306 4.35V, que possúi as seguintes funções:

- Regulador Boost 5V
- Carregador Buck para bateria
- Indicador de Carga
- Detecção inteligente de carga

O Circuito é alimentado externamente com 5 Volts por meio de um conector USB-C, escolhido pela tendência de mercado a virar um conector universal, já estando presente em diversos celulares e aparelhos eletrônicos. Os 5 Volts externos é responsável por fazer a carga da bateria conectada no CI. Quando o CI detecta a fonte externa de alimentação, automaticamente ele realiza um bypass dos 5 Volts para a saída do CI.

Quando a fonte de alimentação externa está ausente, e o CI detecta uma carga na saída do circuito, automaticamente ele troca para o modo de regulador Boost, aumentando a tensão da bateria para 5 Volts e assim alimentando o circuito. Se nenhuma carga é detectada, o circuito entra em standby.

O indicador de carga funciona por meio de 4 LEDs conectados no CI, que acendem ou apagam refente a carga atual da bateria.

A implementação do circuito foi feita baseada no typical application presente do datasheet do componente.

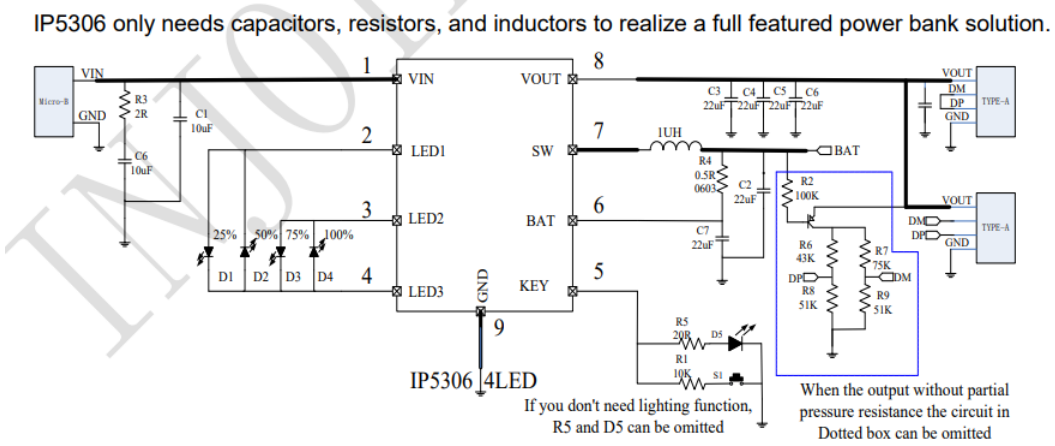

Figura 10: Typical Application do CI de carregamento

O CI possui alguma outras funcionalidades que não utilizaremos, como suporte para 2 saídas e função de lanterna.

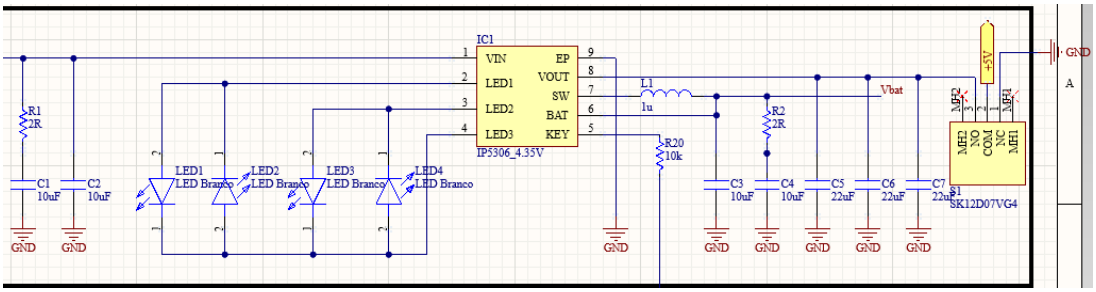

Figura 11: Esquemático do Circuito de Carregamento

5.2.3.3 Regulador 3.3 Volts

Para regular a saída de 5 Volts da alimentação principal da placa, utilizamos um Regulador Linear de 3.3 Volts. O componente escolhido foi o ZLDO1117QG33TA pois o mesmo suportava a corrente máxima que nosso circuito exige. A implementação também foi baseada no typical application do componente.

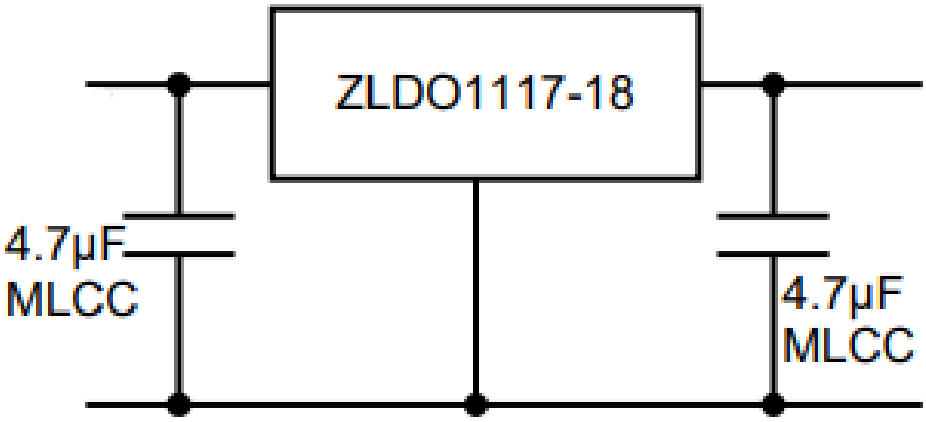

Figura 12: Typical Application do regulador LDO

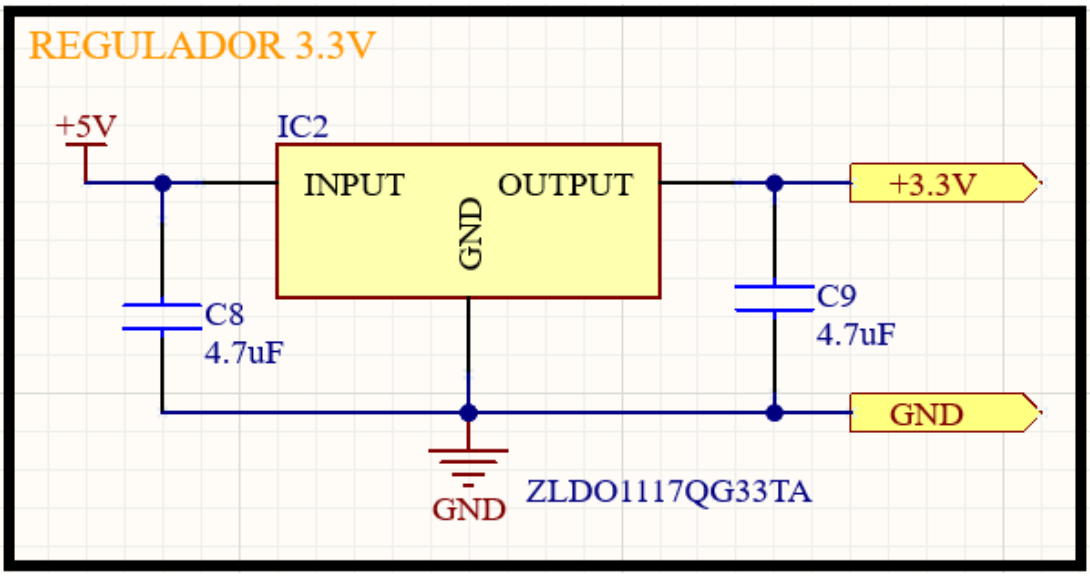

Figura 13: Esquemático do Regulador LDO

5.2.4 Circuito do Microcontrolador

O Microcontrolador escolhido para o projeto foi o STM32U575CIT6, porém devido a falta de estoque no mercado optamos por substituí-lo pelo STM32U575CIT6Q, sua versão com um regulador SMPS interno.

A implementação de seu circuito se deu de acordo com o seguinte esquemático:

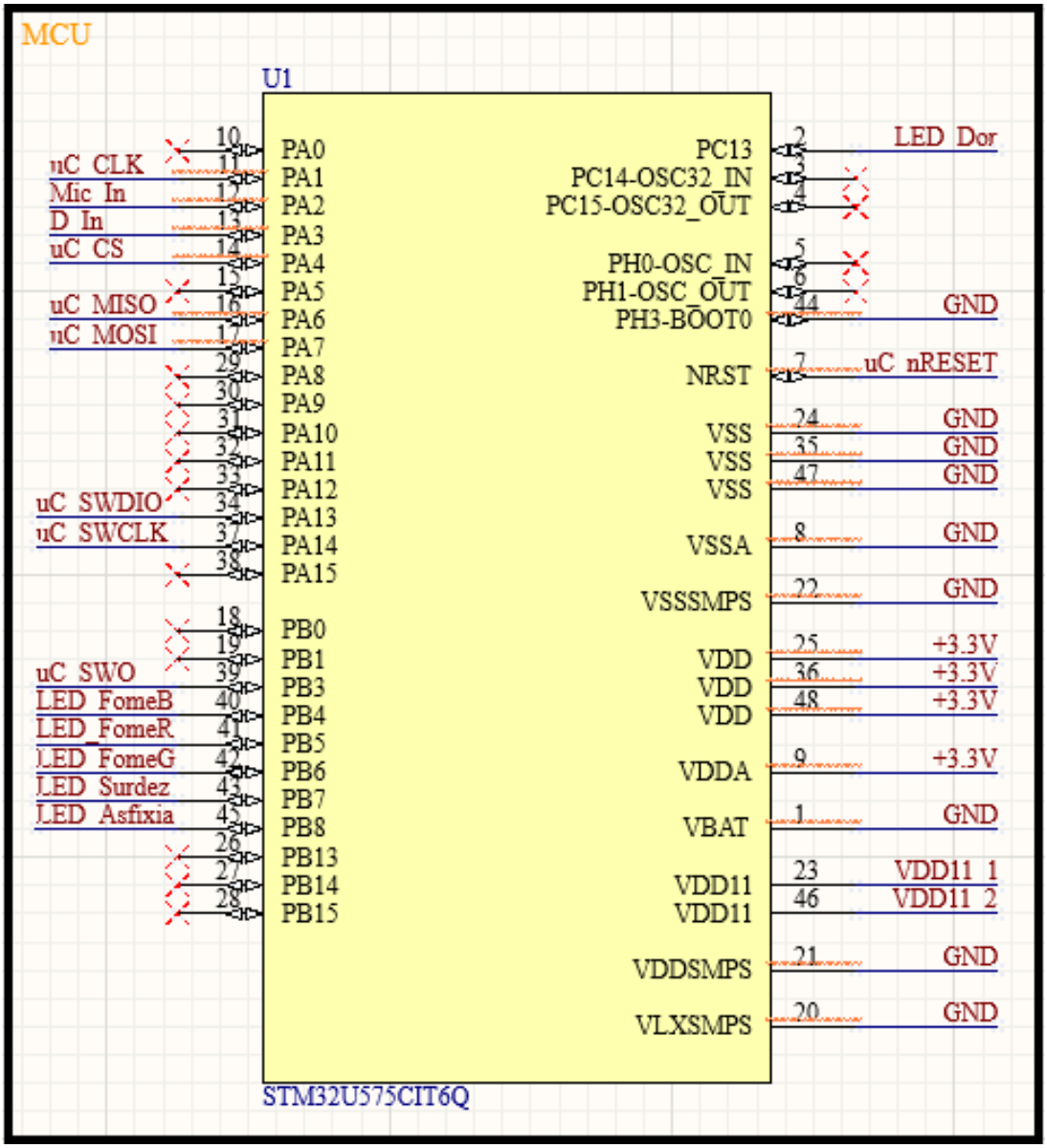

Figura 14: Esquemático Microcontrolador

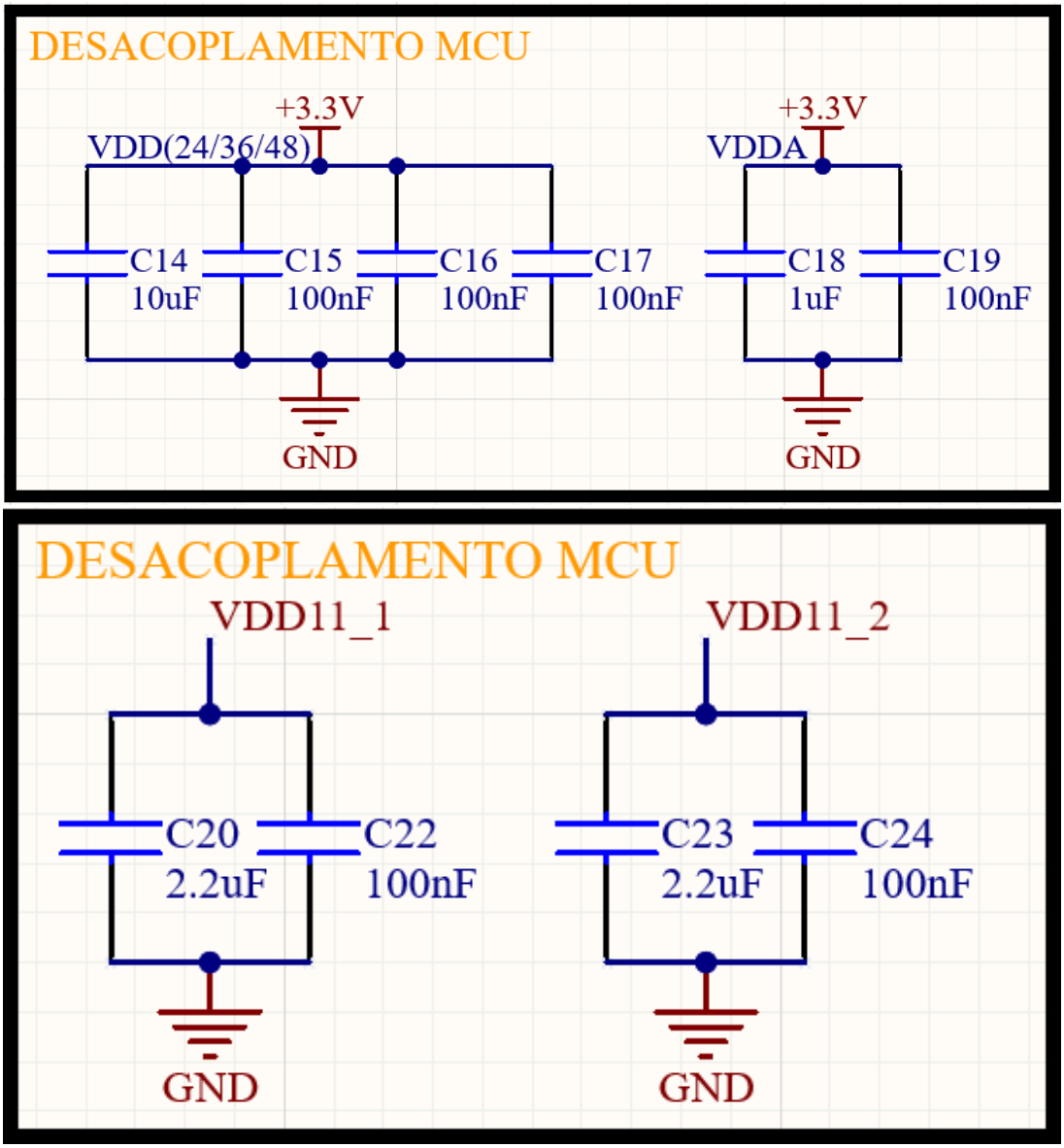

Figura 15: Capacitores de desacoplamento

5.2.4.1 Alimentação do Microcontrolador

Para a alimentação do micro seguimos as seguintes recomendação referenciadas no datasheet disponibilizado pelo fabricante:

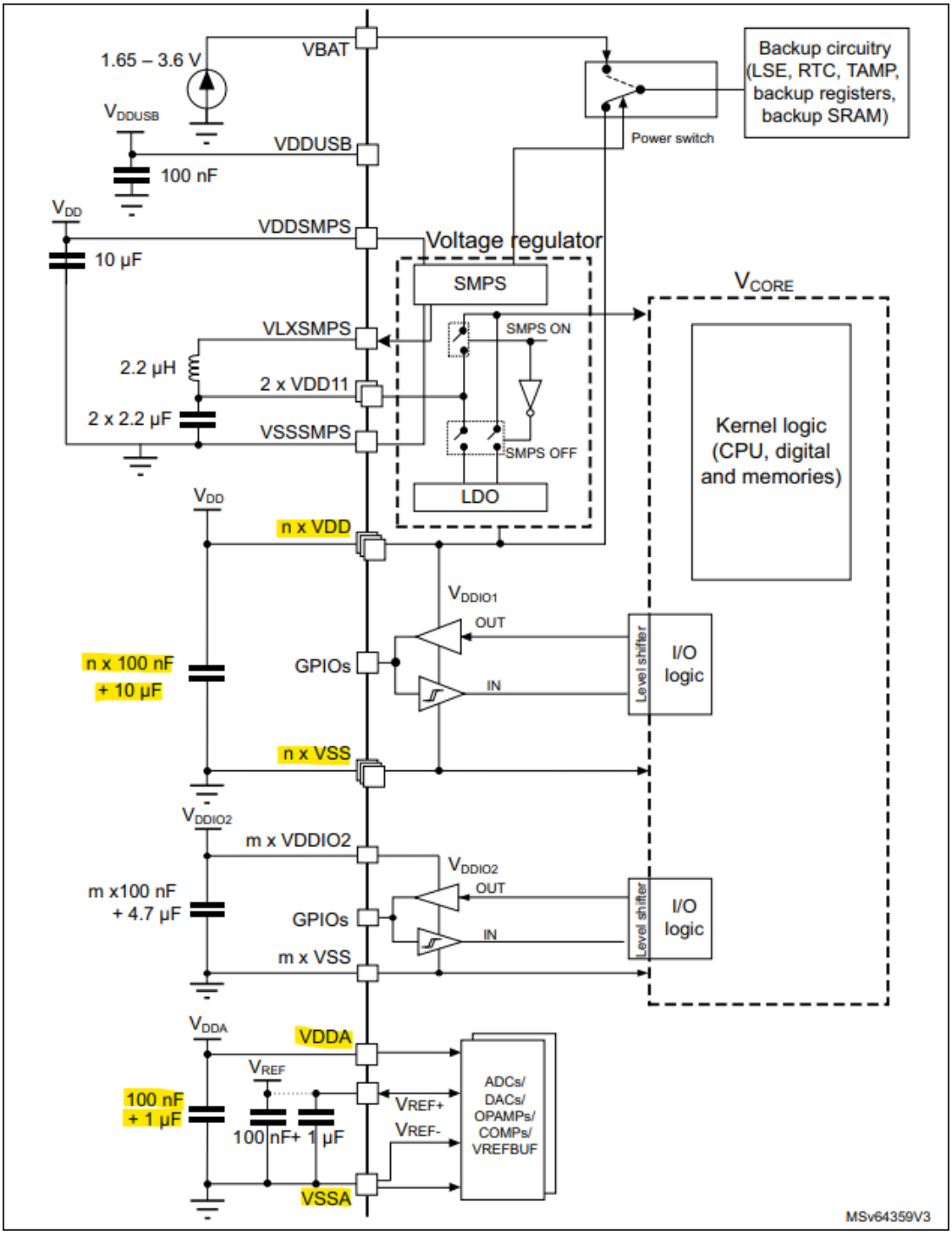

Figura 16: Esquema de alimentação do microcontrolador

Algumas precaução precisaram ser tomadas de acordo com o nosso uso, seguindo as informações presentes no datasheet do componente. O microcontrolador em questão possui um regulador SMPS interno, o qual optamos por não utilizar:

Sobre os pinos do regulador SMPS

"VDDSMPS is the external power supply for the SMPS step-down converter.

It is provided externally through VDDSMPS supply pin. It must be connected to the same supply VDD pin when the SMPS is used in the application. When the SMPS is not used, it is recommended to connect both VDDSMPS and VLXSMPS to GND."

Sobre os pinos VDD11

"SMPS and LDO regulators provide, in a concurrent way, the VCORE supply depending on application requirements. However, only one of them is active at the same time. When SMPS is active, it feeds the VCORE on the two VDD11 pins supplied by the filtered SMPS VLXSMPS output pin. A 2.2 µH coil and a 2.2 µF capacitor on each VDD11 pin are then required. When LDO is active, it supplies the VCORE and regulates it using the same decoupling capacitors on VDD11 pins. It is recommended to add a decoupling capacitor of 100 nF near each VDD11 pin/ball, but it is not mandator."

5.2.4.2 Escolha de pinos

Para os pinos de acionamento dos LEDs escolhemos pinos com porta GPIO e que estivessem próximos para facilitar no projeto do circuito impresso.

Para a entrada do microfone foi escolhido um pino que possuísse uma porta com ADC (Analog Digital Converter) para poder realizar a amostragem do sinal sonoro.

Para a entrada do circuito saturador foi escolhido um pino que possuísse uma porta GPIO capaz de gerar interrupção quando operando em modo Low Power.

E para a interface com a ESP, escolhemos os pinos do periférido SPI do microcontrolador.

5.2.4.3 Interface de Programação

A interface de programação de Debug escolhida foi a SWD (Serial Wire Debug), pelos motivos de ser bastante usual e conhecida e por possuir uma quantidade de pinos necessários menos que a interface J-Tag.

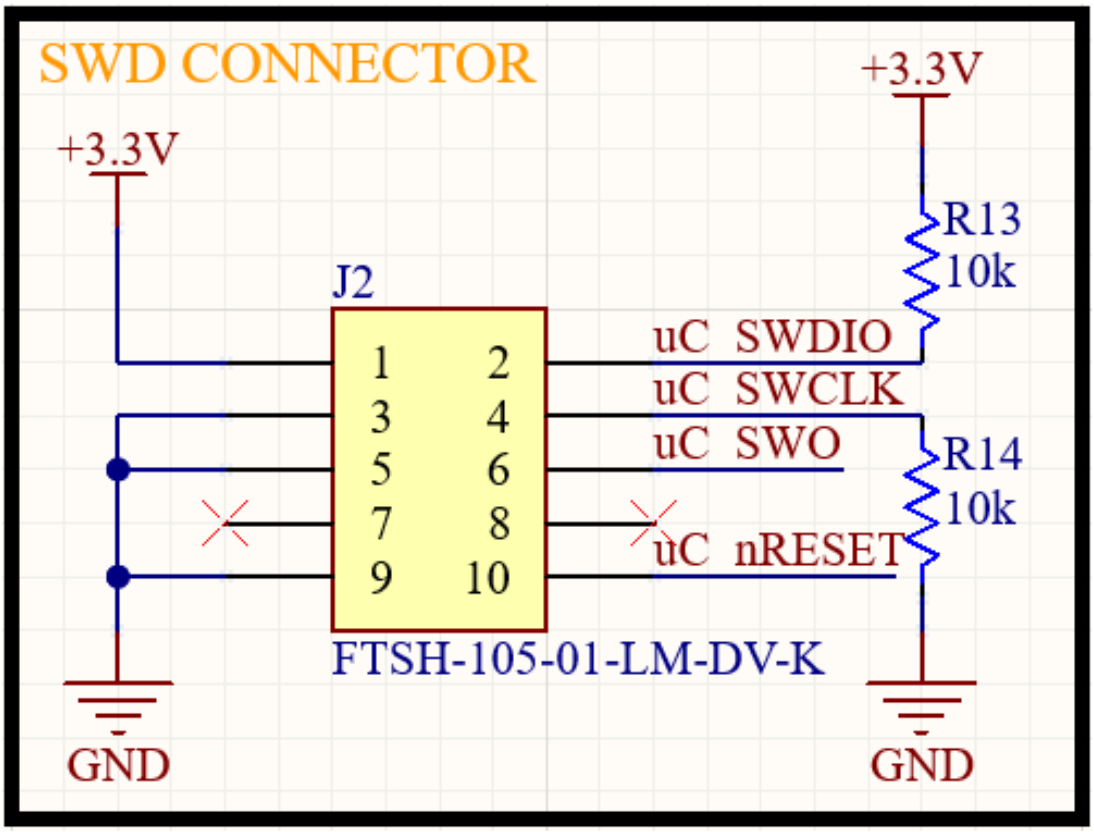

Figura 17: Esquemático da Interface de Programação

5.2.5 Design da PCB

Para a PCB, optamos por desenvolver uma placa de 4 camadas para termos mais flexibilidade na hora de fazer as conexões entre os componentes.

A camada de baixo foi dedicada a comportar todos os componentes relacionados a alimentação do circuito.

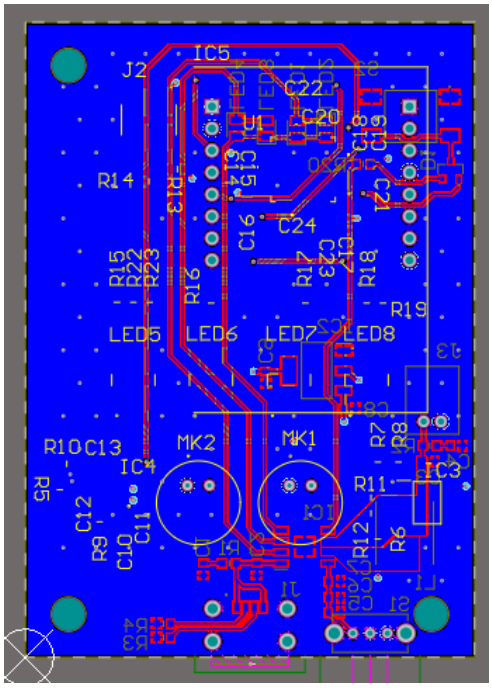

Figura 18: Bottom Layer

Já nas camadas internas deixamos os planos de alimentação (5 Volts e 3.3 Volts), assim como o plano de Terra.

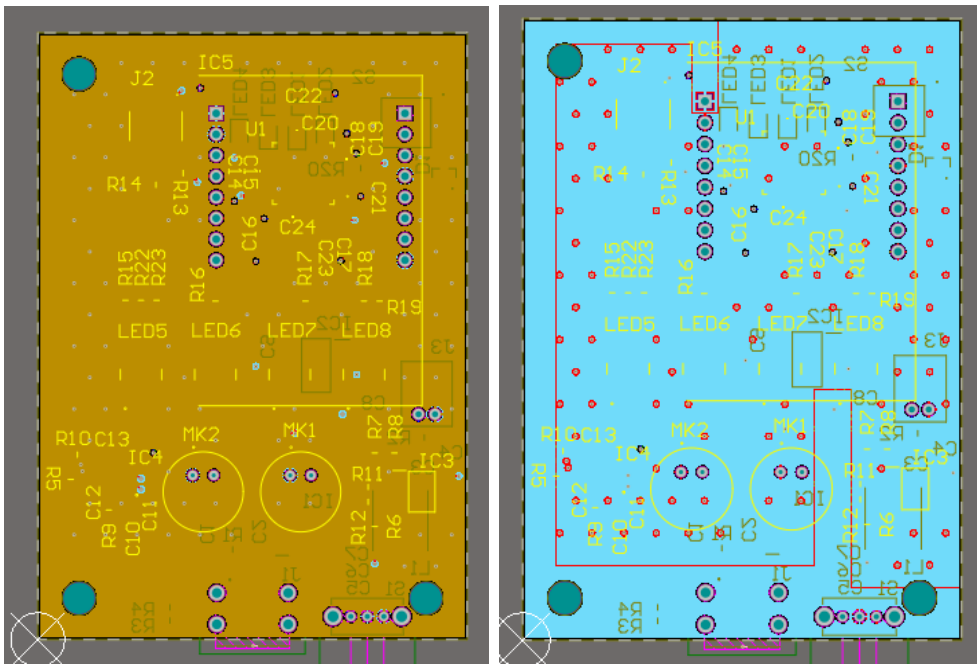

Figura 19: Camadas Intermediárias

Na camada superior deixamos alocados todos os outros componentes.

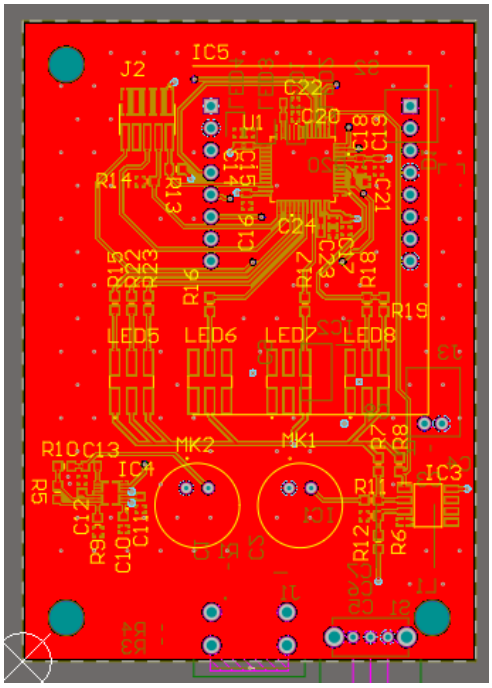

Figura 20: Top Layer

O posicionamento dos componentes também foi pensado de forma a colaborar com o projeto mecânico, já pensando no produto final. Portando o alinhamento dos Leds foi feito, assim como de ambos microfones.

Também foram adicionados 3 furos para fazer a fixação da placa na peça mecânica.

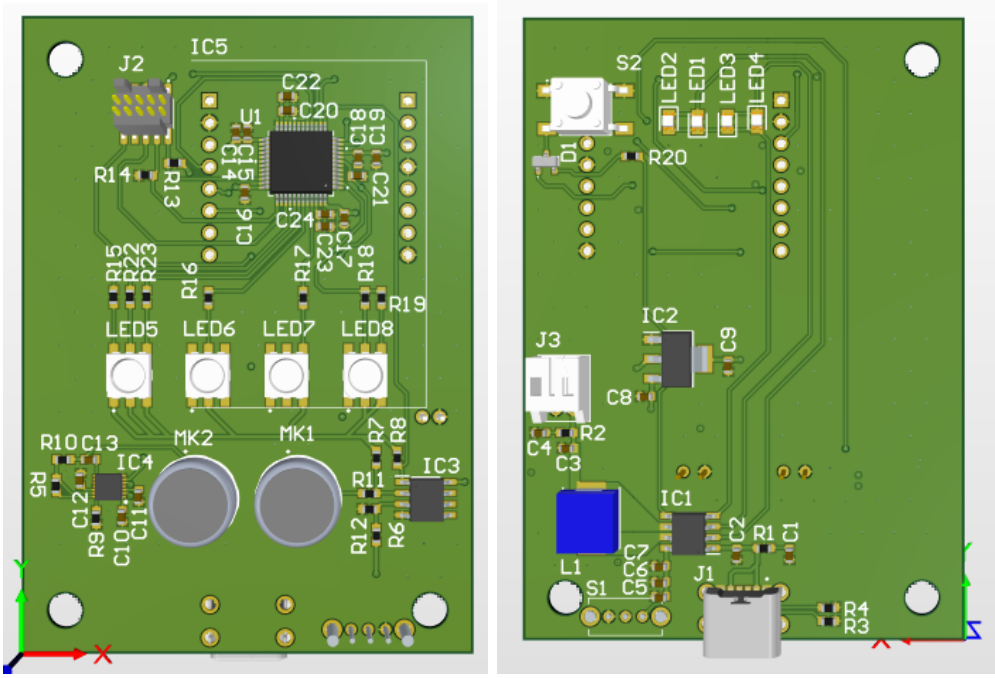

Figura 21: Visualização da Placa em Software

5.3 Projeto de Software

5.3.1 Pré processamento

Definiu-se que o sinal de áudio seria amostrado em **22.05 kHz**. Esse valor foi obtido levando em conta a distribuição espectral de potência média das amostras sonoras contidas no dataset (**22**). A partir do *teorema de Nyquist*, sabe-se que conseguimos amostrar frequências de até 11.03 kHz, que como observado, é suficiente para uma leitura satisfatória das amostras.

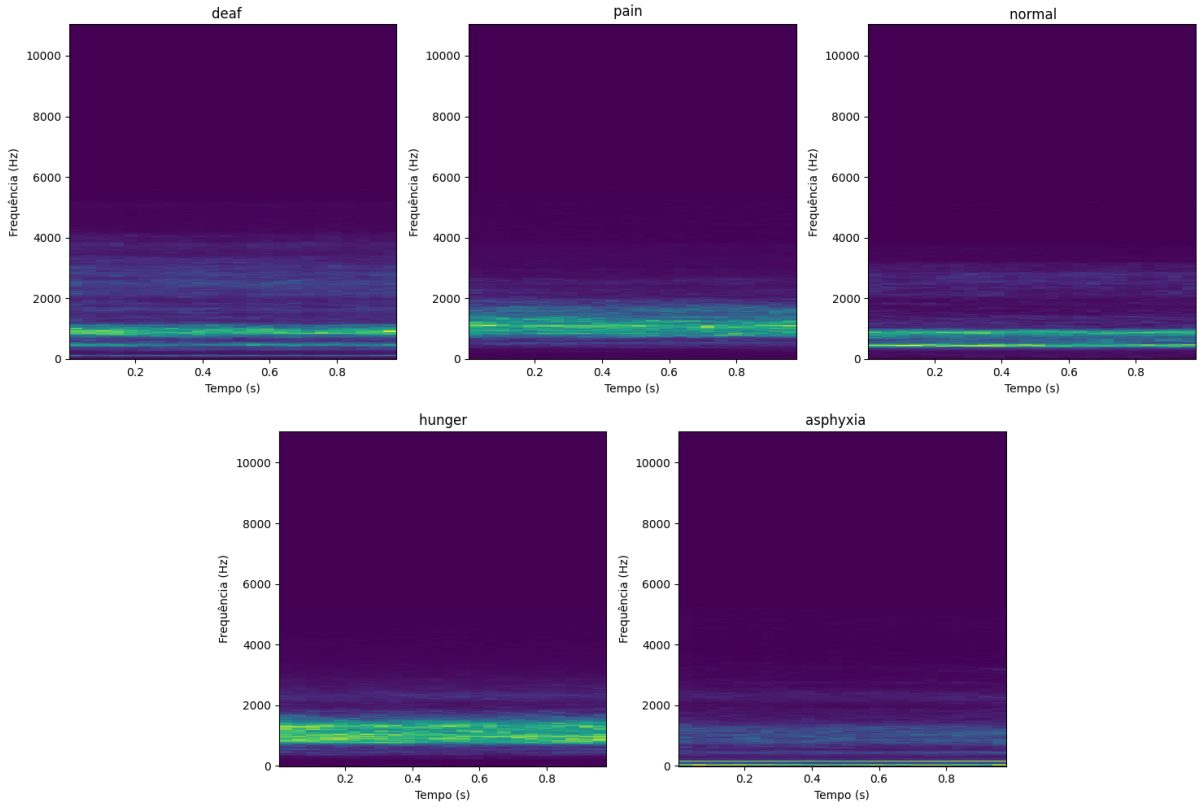

Figura 22: Média dos espectrogramas sobre os dados de treinamento.

Dessa forma, o período das amostras foi definido para **1 segundo** levando em conta a memória que seria consumida do dispositivo para seu armazenamento. Como cada dado de áudio é armazenado em variáveis de 2 Bytes, foi calculado que seria necessário 44,07 KB de memória RAM no dispositivo.

A definição dos parâmetros de calculo do espectrograma foi feita buscando um equilíbrio entre resolução da imagem e tamanho do dado, visto que um espectrograma com menor janela de amostragem apresenta grande resolução, mas gera uma matriz de maiores dimensões. Esse tamanho é prejudicial, visto que aumenta o número de entradas da rede neural, prejudicando assim seu desempenho e consumo de memória.

Assim, foram definidos experimentalmente os valores:

Parâmetro	**Valor**
Tamanho da FFT	1024
Tamanho de HOP	128
Janela	*Boxcar*

Tabela 3: Parâmetros de espectrograma para pré-processamento

Dessa forma as dimensões do espectrograma resultante podem ser calculadas:

$$frames = \frac{samples}{HOP - FFT_{size}} = \frac{22050}{1024 - 128} = 25$$

$$frequencias = \frac{FFT_{size}}{2} = \frac{1024}{2} = 512$$

Portanto, o espectrograma tem dimensão **25x512**, fazendo com que a rede neural tenha 12800 valores de entrada.

A última etapa de processamento é a normalização, onde a matriz resultante é dividida por um valor escalar de forma a manter sua norma euclidiana unitária. Essa etapa é importante para que a classificação não sofra tanto impacto com a amplitude do som, consequência da proximidade do dispositivo ao recém nascido.

5.3.2 Modelo de classificação

Existem diversos modelos adaptados para a execução em dispositivos móveis ou embarcados. Foi realizada a análise desses modelos, com grande foco no consumo de memória não volátil. Os valores de consumo encontrados (**Tabela 5.3.2**) inviabilizariam a sua implementação utilizando microprocessadores avaliados, que continham alguns KB de memória flash. Dessa forma, foi decidido partir para o desenvolvimento de um modelo próprio que cumprisse tais requisitos.

Rede Neural	Parâmetros Treináveis (milhões)	Memória Flash Necessária*
MobileNet v3	3.2	12 MB
MobileNet v2	2.3	8,7 MB
MobileNet v1	2.9	11,1 MB
EfficientNet v2	4	15 MB

Tabela 4: Consumo de memória de outros modelos

* Supondo parâmetros armazenados em variáveis float32

Por se tratar de um modelo para a classificação de áudios, optou-se por um modelo baseado RNNs, em particular, LSTMs, pelo seu frequente uso em machine learning aplicado a audios observado na literatura [colocar referencia aqui]. A escolha pela LSTM também se deu pela possível vantagem que células de memória adicionariam ao modelo, ao analisar choros de asfixia, que se destacam por ser composto por pequenos trechos de choros pausados, dado a falta de ar do recém nascido. Dessa forma, as unidades de memória podem utilizar esses dados passados para identificar as pausas de som, e assim auxiliar a classificação.

Dessa maneira, o modelo é composto por uma camada LSTM, seguida por uma camada densa, que enfim classifica o dado de entrada dentre 5 classificações possíveis.

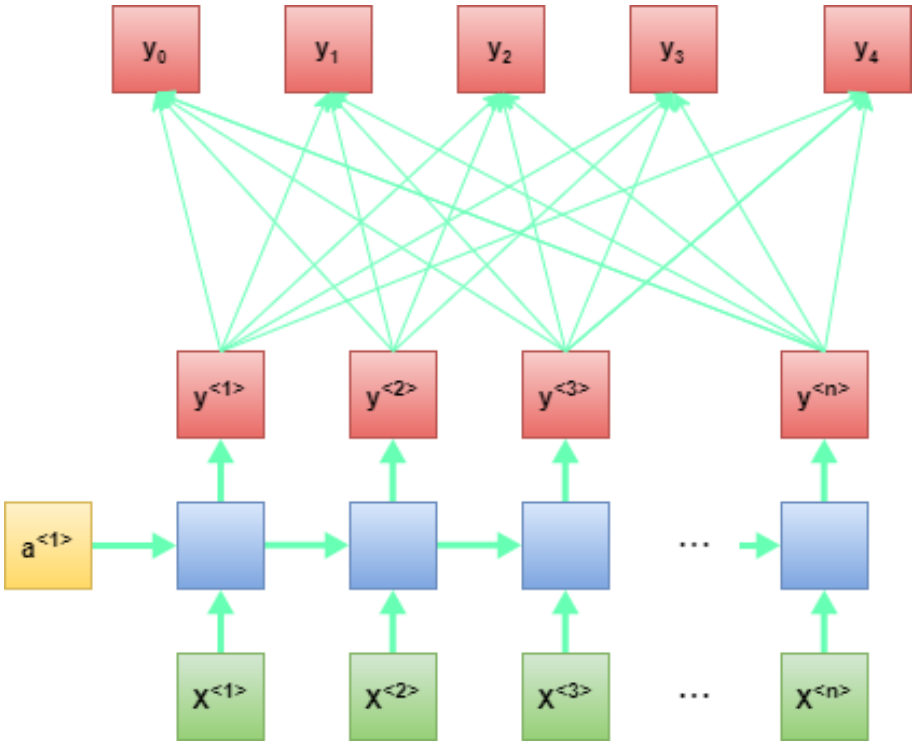

Figura 23: Arquitetura de rede inicial.

Além disso, foi acrescentada uma camada de *Dropout* antes da camada densa, de forma a combater *overfitting*.

Visto que a rede tem como objetivo sua aplicação em um dispositivo embarcado, o seu consumo de memória volátil e não volátil é um indicador importante para o projeto. Para fazer essa avaliação, foi utilizado o pacote *X-CUBE-AI*, para a conversãodo modelo de rede neural desenvolvido para a linguagem C.

Com isso, foi obtido que a rede utiliza 597.38 KiB de memória Flash, sendo grande parte destinada a pesos, e 51,67 KiB de memória RAM **Figura 24**. Dessa forma, o modelo de rede neural foi um grande definidor de requisitos de hardware, principalmente quanto à capacidade da memória Flash.

```
Total Flash:      611716 B (597.38 KiB)
    Weights:      593940 B (580.02 KiB)
    Library:      17776 B (17.36 KiB)
Total Ram:        52908 B (51.67 KiB)
    Activations:  51296 B (50.09 KiB)
    Library:      1612 B (1.57 KiB)
    Input:        49248 B (48.09 KiB included in Activations)
    Output:       20 B (included in Activations)
```

Figura 24: Análise de consumo de memória da rede neural.

5.3.3 Dados de treino

Alguns datasets são comumente utilizados na literatura na área de classificação e identificação de choros. Para escopo deste trabalho, foram filtrados os datasets que tinham suas amostras classificadas quanto a causa do choro. Restaram então bases como SPLANN, Baby2020, Baby Chillanto e outras iniciativas privadas de coleta colaborativa de dados.

O dataset Baby Chillanto foi o utilizado para o trabalho pela maior facilidade de contato com os pesquisadores envolvidos. A Base é dividida entre as classes **Fome**, **Dor**, **Asfixia**, **Surdez** e **Normal**, distribuidas conforme **Figura 25**.

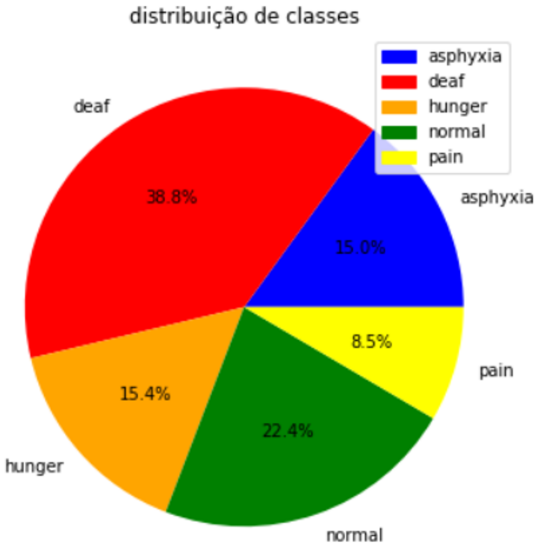

Figura 25: Distribuição de Classes no Dataset Baby Chillanto.

As amostras são dados sonoros de 1 segundo de duração, retiradas de áudios maiores gravadas e classificada com ajuda de um conjunto de médicos, é possível observar que algumas classes foram amostradas de mais indivíduos que outras.

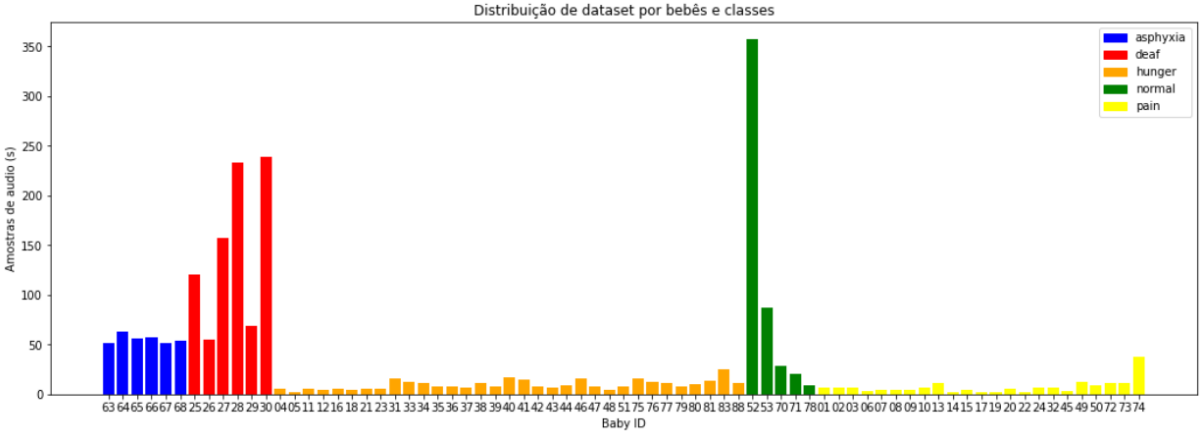

Figura 26: Distribuição de Classes por indivíduo recém nascidos.

5.3.4 Data Augmentation

Para aumentar a capacidade de generalização da rede neural, foram aplicados algoritmos de *Data Augmentation* sobre os dados de treinamento, responsáveis por criar dados novos, baseados nos já existentes, com algumas alterações. Dessa forma, além de aumentar a capacidade de generalização do modelo, é possível prevenir *overfitting* e acrescentar mais dados de classes menos numerosas, tornando assim o banco de dados mais equilibrado.

Foram feitas mudanças mais simples, e outras mais complexas aos dados durante o processo. Dentre as mais simples, foram realizadas cortes aleatórios nos áudios originais (completos, sem cortes) e mudança na velocidade de reprodução sem mudança das frequências. Estudos apontam que mudar as frequências dos dados não tem grande influência no treinamento.

Para perturbações mais complexas nos dados, foi utilizada a biblioteca *pedalboard* da empresa Spotify, reconhecida pela mesma como uma ferramenta que "faz o processo de 'Data augmentation' muito mais rápido e produz resultados mais realistas". Dela, foram utilizadas funções de *Reverb* e *Distortion*, funções julgadas capazes de simular uma mudança no ambiente de gravação e adicionar possíveis ruídos.

Todas as mudanças tiveram sua intensidade (grau de distorção, por exemplo) aleatorizados, e aplicados à amostras aleatórias do conjunto de dados originais, permitindo eventualmente a combinação desses efeitos.

Os valores de amostras geradas foram escolhidas de acordo com cada classe, almejando deixar o conjunto de dados o mais equilibrado o possível, facilitando o posterior treinamento da rede (**Figura 27**).

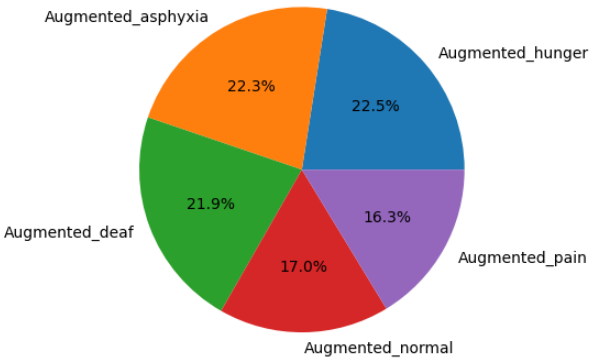

Figura 27: Distribuição de Classes no Dataset após data augmentation.

5.4 Projeto de Firmware

5.4.1 Arquitetura

O Firmware foi projetado conforme uma arquitetura de camadas, de forma a organizar a forma como as funcionalidades seriam implementadas (**Figura 28**). Na camada de mais baixo nível, a camada de abstração de hardware "HAL"é responsável por se comunicar

diretamente com os periféricos. Neles, estão também as configurações de inicialização de parâmetros essenciais como frequência de clock, taxas de amostragem e configuração de pinos de entrada e saída.

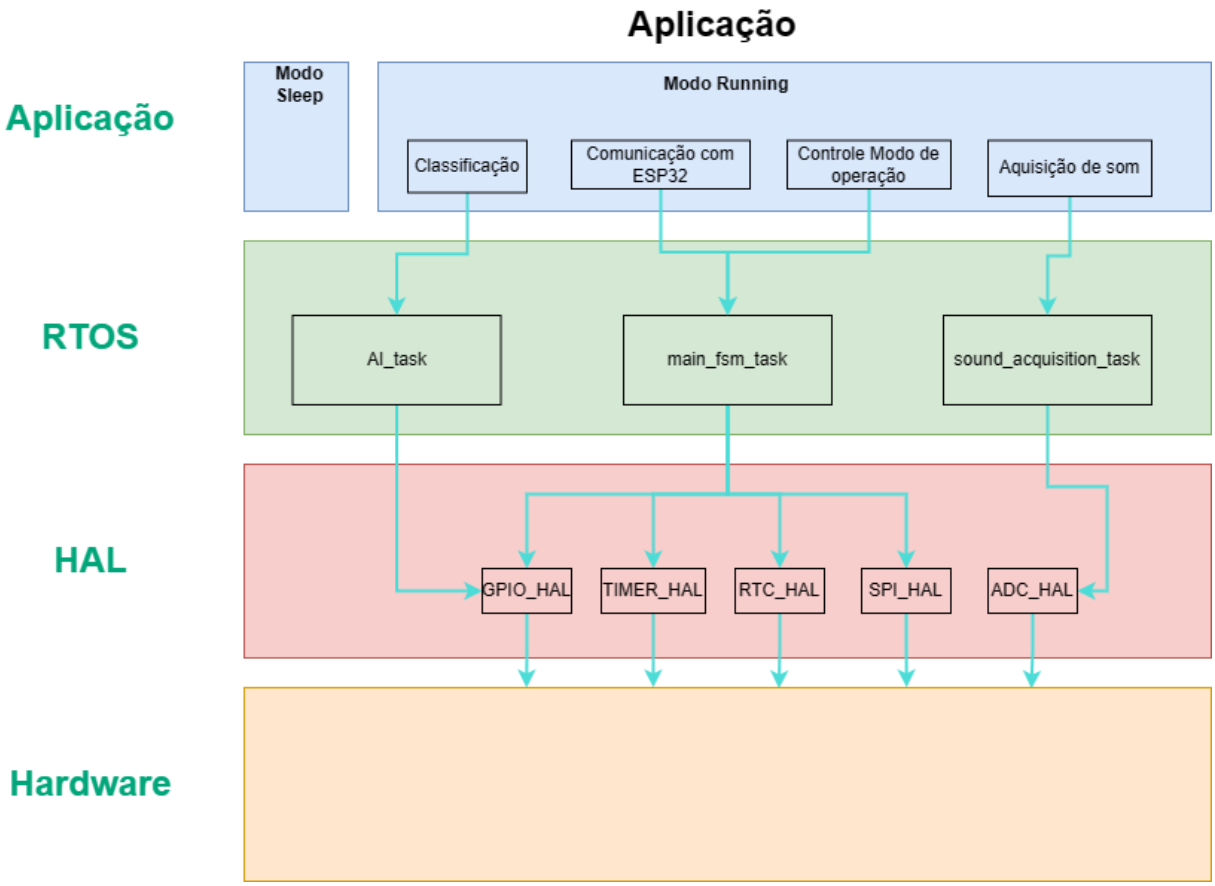

Figura 28: Arquitetura de firmware

Em seguida, para uma implementação mais direta do paralelismo das funcionalidades do dispositivo, foi implementada a camada de sistema operacional de tempo real utilizando FreeRTOS. Nela, foi possível dividir o código em três diferentes *tasks*.

- ***AI_task***- Responsável por executar o pré processamento dos dados de áudio, e a propagação da rede neural. É a *task* mais custosa, tanto em tempo como em consumo de memória.

- ***main_fsm*** - Implementação da Máquina de Estados Finita principal, que controla o fluxo de trabalho do dispositivo, realizando operações como: ativar *Timers* para controle geral, alternar entre diferentes modos de operação e comunicação com ESP32.

- ***sound_acquisition*** - Aquisição de valores analógicos de tensão vindos do microfone, e alocação dos valores em vetores que serão lidos pela *AI_task*.

5.5 Projeto Mecânico

Para a mecânica do produto, foi pensado em um case simples inspirado nos rádios de comunicação Walkie Talkie. Na parte superior, o case conta com furos para liberar a passagem de som para os microfones, assim como furos onde serão encaixados Light Pipes para os LEDs. Também contando com um baixo relevo para colarmos um adesivo referente a identidade visual da marca e do produto.

Já na parte inferior, o case consta com 4 furos para os LEDs de indicação de bateria, assim como um furo para acessar o botão de reset do dispositivo.

A peça também conterá a saída para o conector USB-C e a chave geral da alimentação.

Ainda na parte de trás pensamos em um suporte para que seja possível pendurar o dispositivo no berço da criança. Futuramente esse suporte será substituído por uma presilha capaz de prender em lugares mais variados.

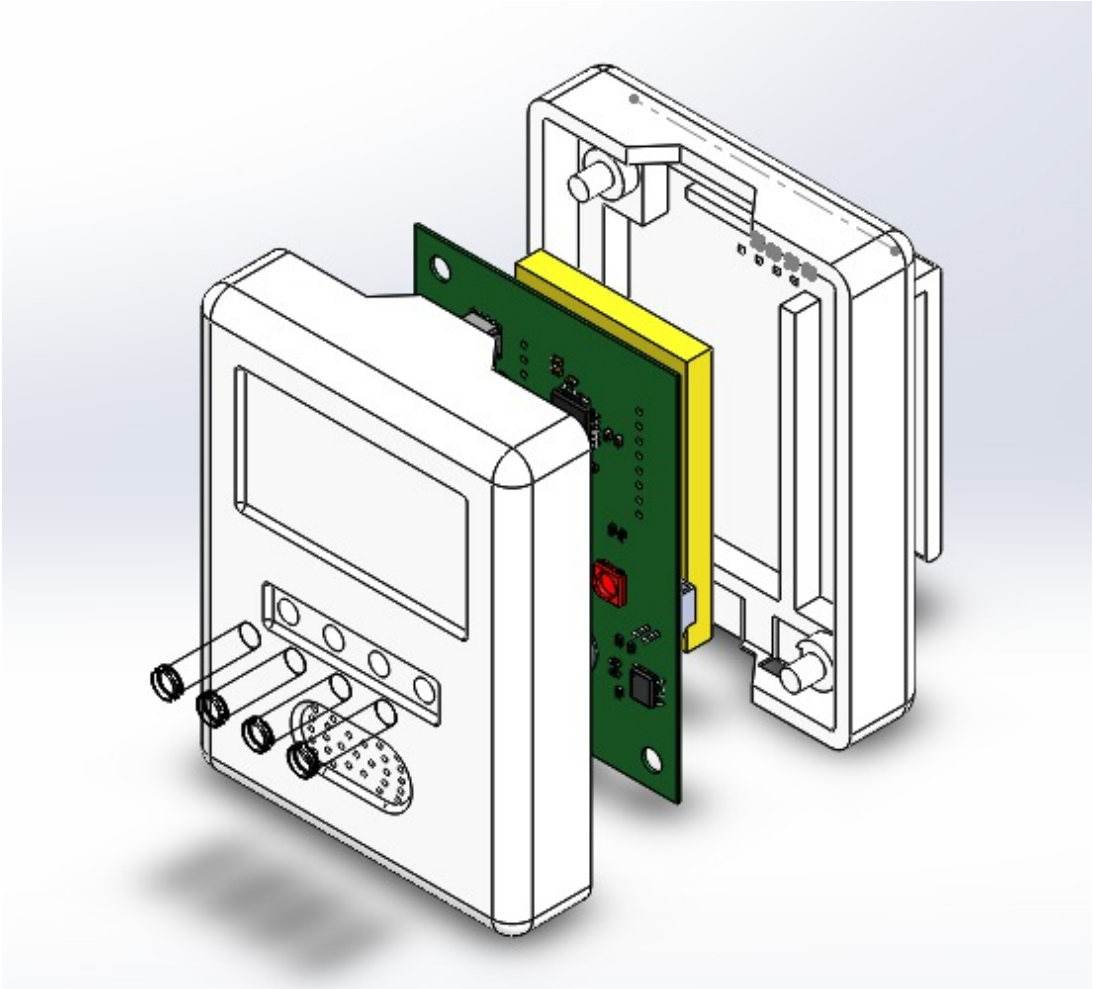

Figura 29: Vista Explodida do Projeto Mecânico

5.6 Implementação do Hardware

5.6.1 Pedido de placas e componentes

Após finalizado e revisado o projeto da PCB, mandamos fabricar a placa no site JLCPCB. Foram fabricadas o total de 5 placas. Já os componentes foram comprados tanto no site LCSC quanto na DigiKey, e foram pedidos componentes suficientes para 2 placas, ou seja, temos 3 placas virgens sobressalentes.

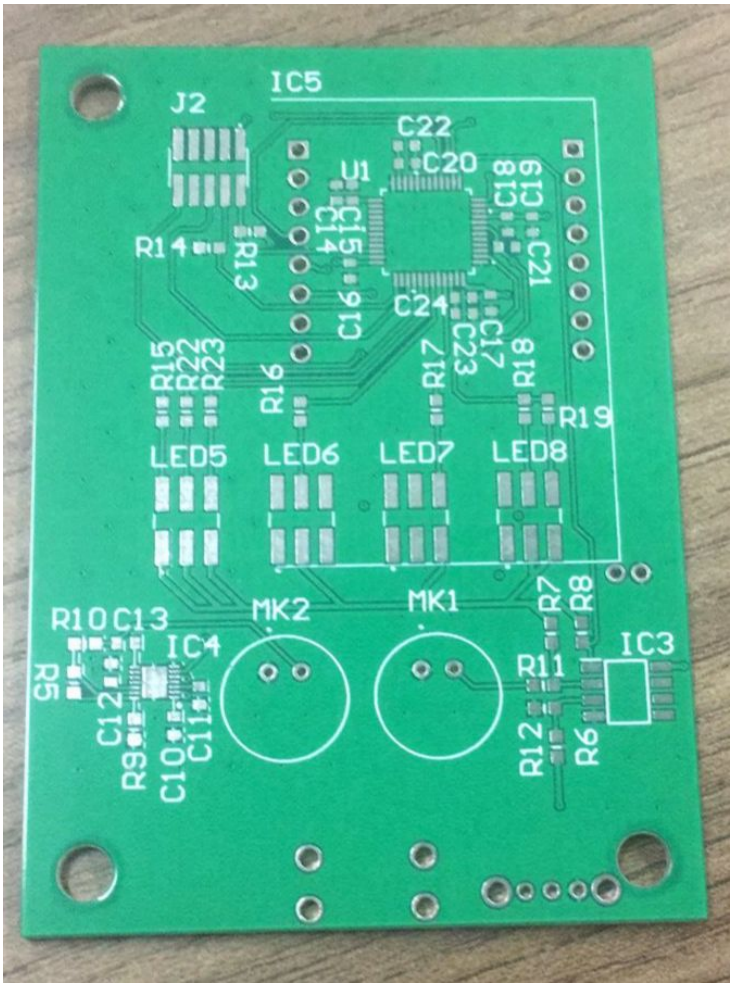

Figura 30: Placa Virgem

5.6.2 Soldagem das placas

Para soldar as placas, optamos por soldar cada conjunto separado para facilitar na depuração do circuito.

Começando com o circuito de alimentação, soldamos os conectores, reguladores e componentes auxiliares como capacitores e resistores. Dessa forma, conseguimos testar

se havia as tensões desejadas em cada ponto do circuito antes de soldar o resto dos componentes, evitando uma possível queima de componentes.

Durante todo o processo de solda, foi verificado continuidade entre os pontos críticos da placa para evitar curtos-circuito.

O próximo módulo a ser soldado foi o circuito saturador. Este foi soldado também sem nenhuma dificuldade.

Em seguida, soldamos o circuito do amplificador do microfone. Esse por sua vez, tivemos grande dificuldade em soldar o CI do amplificador devido ao seu footprint extremamente pequeno, sendo muito complicado soldá-lo sem curtar seus PADs. Após muitas tentativas, acabamos estragando um dos PADs da placa e optamos por soldar o módulo utilizado nos testes por meio de wireups. Dessa forma seria possível prosseguir com o desenvolvimento do protótipo sem perder mais tempo tentando soldar o componente problemático.

Por último, foi soldado o circuito do microcontrolador com os LEDs de identificação e o conector de programação.

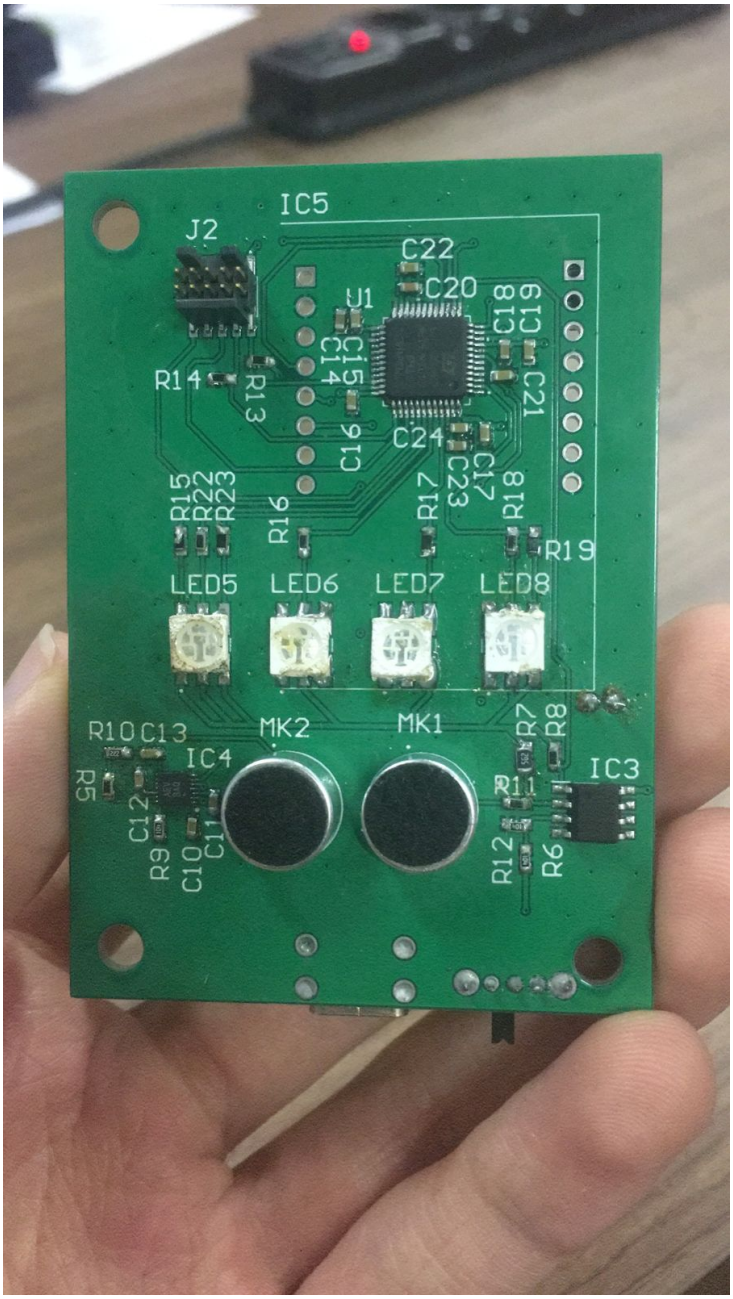

Figura 31: Placa Soldada Completa

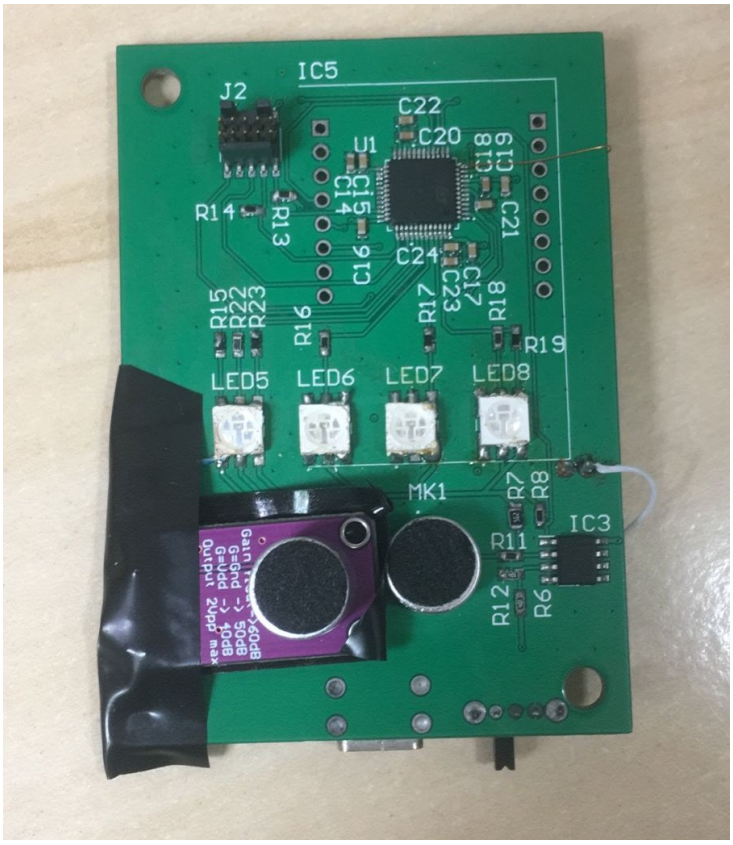

Figura 32: Placa Soldada Com Modulo Externo

5.6.3 Bring Up e Testes dos Módulos

Para testar os módulos da placa, foram feitos alguns códigos individuais.

5.6.3.1 LEDs

Para testas os LEDs, fizemos um código simples que apenas acendia todos o LEDs presentes um por um.

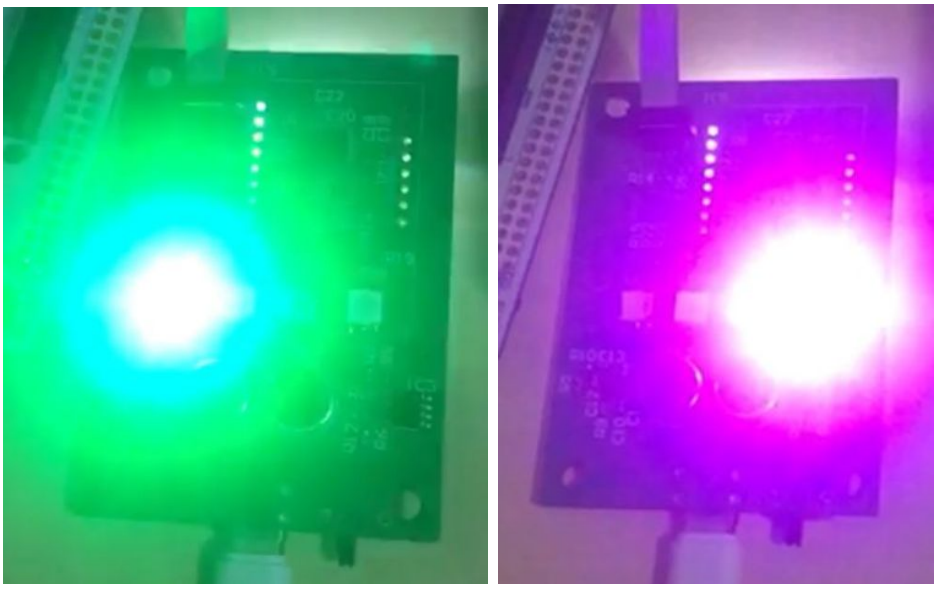

Figura 33: Teste dos Leds

5.6.3.2 Circuito Saturador

Para testar o módulo do microfone saturador de Wake-Up, fizemos um código em que a borda de decida do circuito saturador gera uma interrupção no microcontrolador acendendo um LED. Assim é possível verificar se estamos conseguindo identificar os sons altos.

5.6.3.3 Circuito Amplificador

O teste do circuito amplificador foi executado realizando uma amostragem a 22 kHz durante 1 segundo e mandando as informações para o software CubeMonitor. Dessa forma foi possível plotar o sinal adquirido e converte-lo em um arquivo de áudio. Foi observado uma grande quantidade de ruído provavelmente originário das trilhas remanescentes na PCB após o WireUp do módulo externo. Porém, ao executar o processamento do sinal, foi observado que o ruído não interfere tanto no espectrograma e consequentemente na classificação da rede.

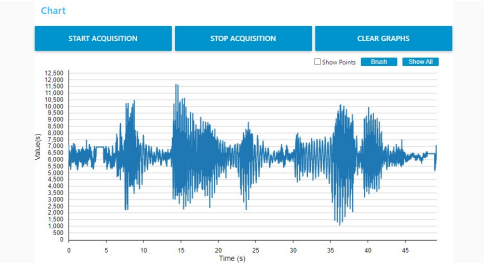

Figura 34: Aquisição de Áudio pela placa

5.7 Implementação de Software

5.7.1 Treinamento

O treinamento foi realizado sobre as 4220 amostras do banco de dados após o processo de *data augmentation*. Foram separadas 30% dessas para compor o conjunto de validação. Em seguida, foi realizado o treinamento por 40 *epochs*, utilizando *Bach size* de 64 amostras.

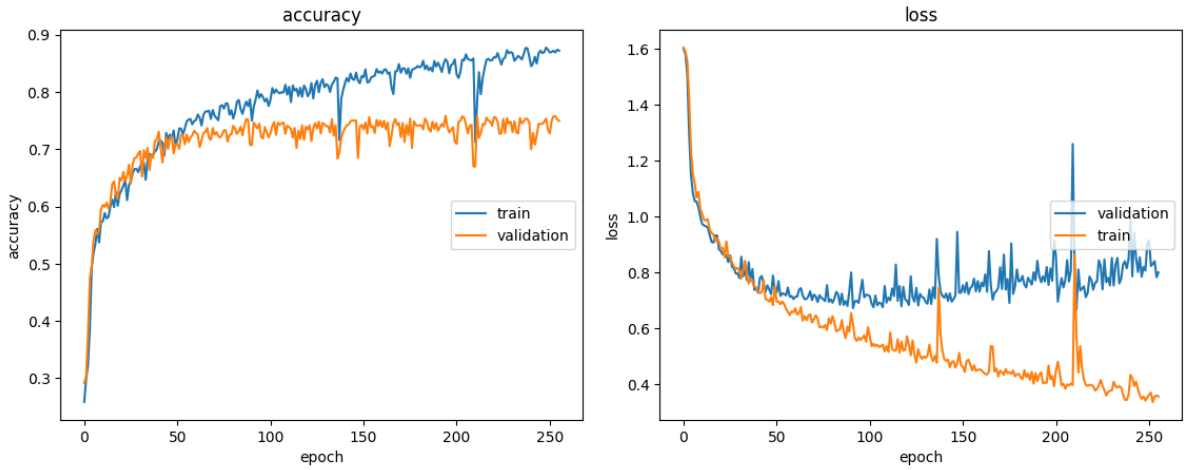

Figura 35: Função de custo e acurácia ao longo do treinamento

Em seguida, foi gerada a matriz de confusão para o modelo, classificando os dados de validação e comparando com suas respectivas classes.

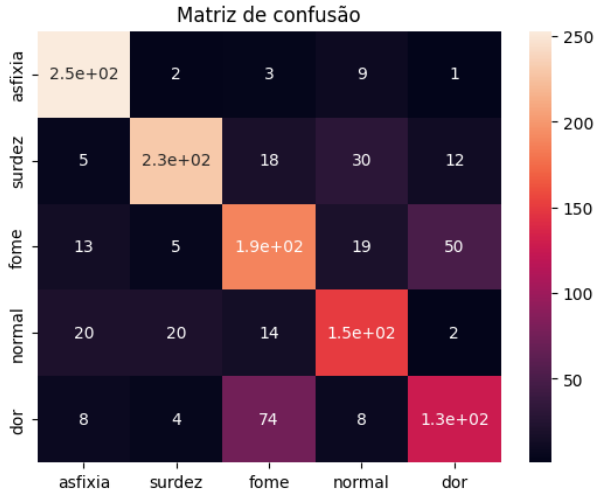

Figura 36: Matriz de confusão calculada sobre os dados de validação.

A matriz demonstra que foi atingida **uma precisão de 75%**, valor coerente com os obtidos no gráfico de treinamento. Também conclui-se que o modelo classifica com maior

facilidade amostras da classe asfixia e surdez. Por outro lado, áudios de fome e dor são confundidos entre si.

Para uma melhor visualização do resultado, foi aplicado o modelo sobre um áudio original (sem cortes), e foi possível observar os resultados comentados.

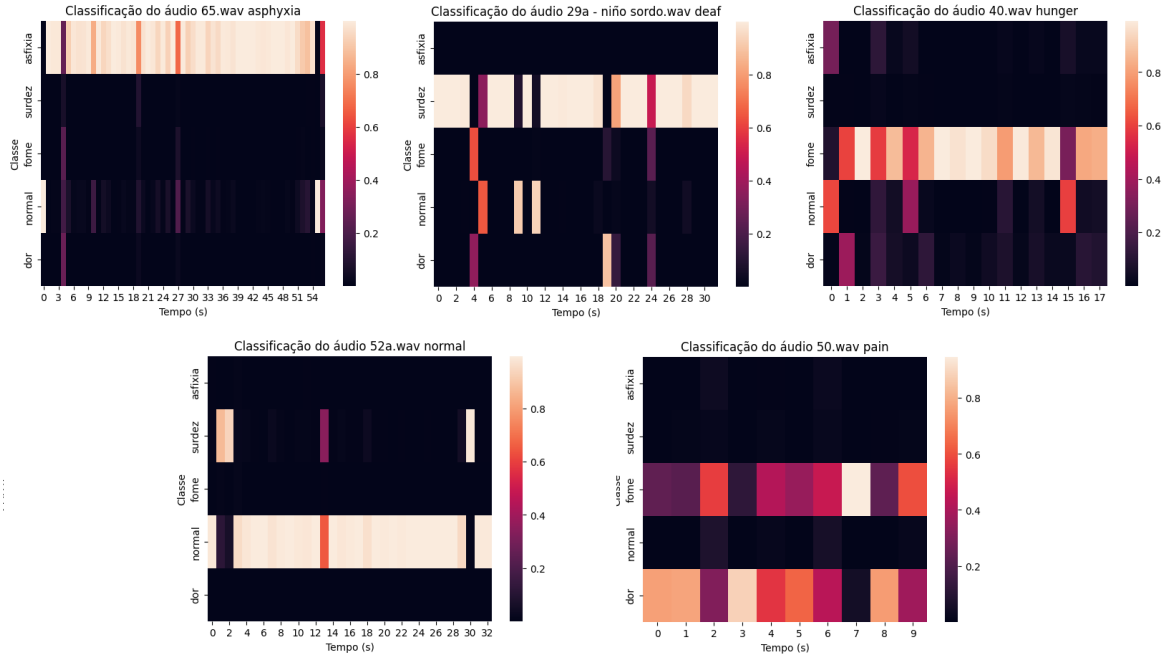

Figura 37: Classificação de arquivos completos, separados em amostras de um segundo

5.8 Implementação do Firmware

5.8.1 Aquisição de som

A leitura do analógica do valor de tensão no pino do microfone é feito utilizando o Conversor Analógico Digital. Segundo dados do microcontrolador escolhido, o ADC utilizado apresenta taxa de amostragem máxima de **2.5 MHz**, superior à de 22.05 kHz definida como requisito. Além disso, a resolução da medição é de 14 bits. Levando em consideração a faixa de medição de 0 - 3.3 V, calcula-se uma resolução de leitura de **0.2 mV**.

A taxa de amostragem obtida pode ser calculada por meio da equação:

$$f = \frac{f_{adc}}{n_{cycles}}$$

Onde f_adc representa a frequência de clock do ADC, e n_cycles o número de ciclos do periférico que serão utilizadas para cada leitura. Visando uma maior precisão de leitura, buscou-se um valor máximo de número de ciclos viável, obtendo então o valor de **391.5 ciclos/leitura**. Com esse valor, foi possível calcular o valor necessário de f_adc.

$$f_{adc} = f \times n_{cycles} \approx 8.6MHz$$

Para ajustar alcançar a frequência de amostragem desejada, é necessário configurar os divisores de clock do processador. O software STM32CubeMX foi utilizado para o cálculo automático desses parâmetros (**Figura 38**).

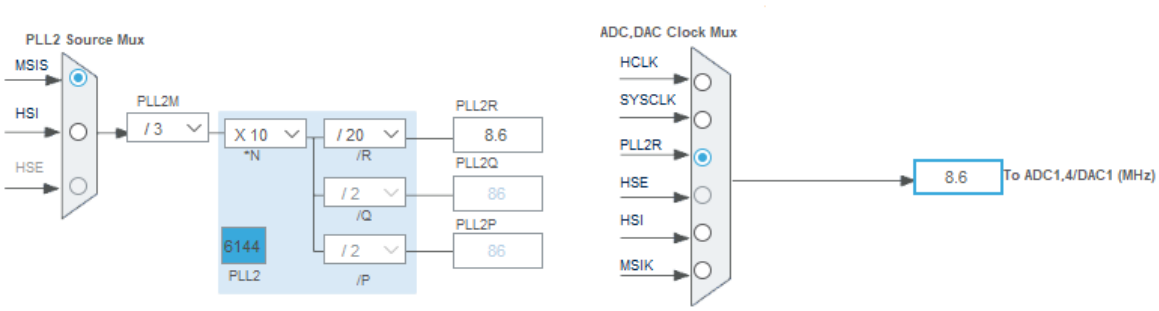

Figura 38: Divisores e multiplexadores para obtenção de clock de ADC de 8.6 MHz.

Para validação da frequência de amostragem, foi utilizado um *Timer* para a medição do tempo necessário para preencher um vetor de 22050 amostras. O valor observado dessa variável foi de 1030 ms, o que demonstra uma frequência de amostragem de 21.4 kHz. Além disso, foi gerado o gráfico das amostras coletadas ao aproximar um som de 400 Hz próximo ao microfone (**Figura 39**).

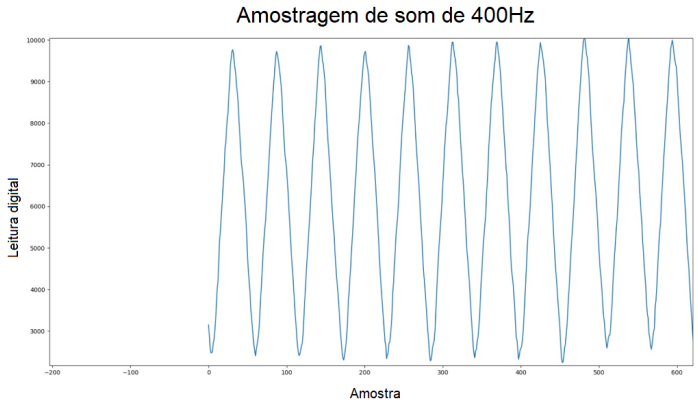

Figura 39: Visualização de sinal amostrado. Som de 400 Hz próximo ao microfone do dispositivo.

5.8.2 Pré-processamento

No ambiente de treino da rede neural, o espectrograma era obtido utilizando funções da biblioteca *scipy*. Era necessário, então, implementar funções em firmware capazes de gerar espectrogramas de forma bastante semelhante aos recebidos pela rede neural durante o seu treinamento. Além disso, por se tratar de uma operação feita exaustivamente pelo dispositivo, deve ser uma função computacionalmente eficiente.

Dessa forma, foi utilizada funções CMSIS de operações vetoriais e processamento digitais de sinais, tirando assim maior proveito da arquitetura do processador e suas instruções.

Primeiramente, foi feita uma comparação entre a FFT embarcada e do *scipy*, visto que é a função básica de geração do espectrograma (**Figura 40**). Notou-se que a FFT calculada no dispositivo apresenta visivelmente maior ruído. Uma possível explicação para esse comportamento é a diferença entre tipos de dados utilizados: enquanto o scipy utiliza variáveis flutuantes de 64 bits para seus cálculos, o firmware foi implementado utilizando variáveis inteiras de 16 bits, tendo assim menor resolução.

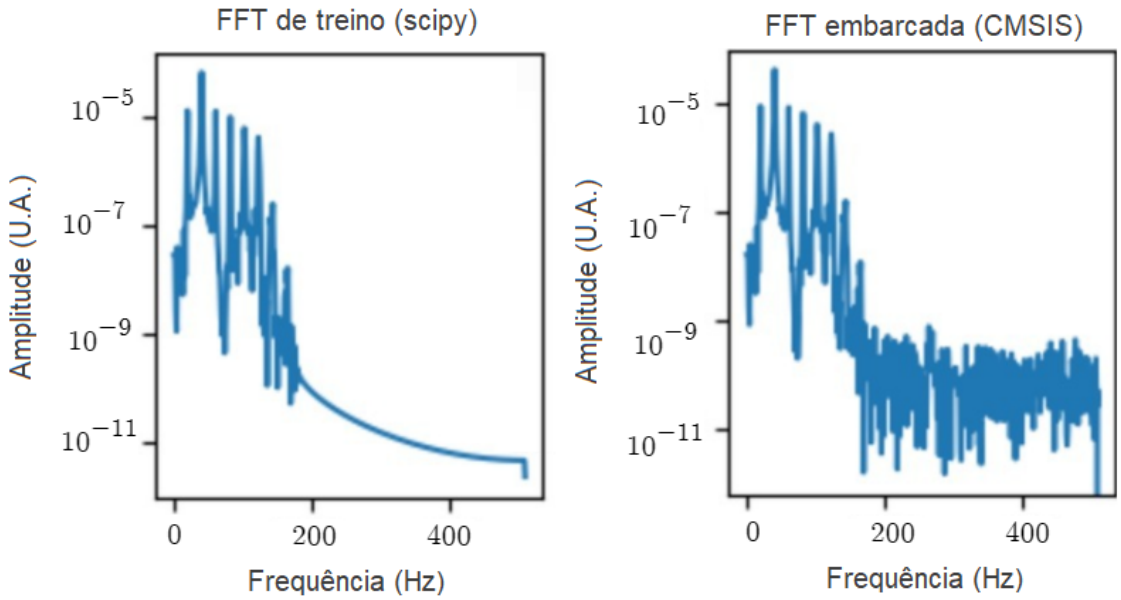

Figura 40: Comparação entre FFTs geradas durante o treinamento e execução no dispositivo embarcado.

5.8.3 Rede Neural embarcada

A biblioteca em C contendo as funções e parâmetros da rede neural foi gerada pelo X-CUBE-AI a partir do modelo gerado pelo Keras. Para verificar se a implementação

foi feita corretamente, foi utilizado depurador para ler o resultado da propagação da rede neural, e comparada com a obtida pelo modelo de treinamento em Python, utilizando os mesmos dados de entrada.

Além disso, para verificar os dados do pré-processamento embarcado, foi criado um código de testes, que envia a matriz contendo o espectrograma via protocolo serial do dispositivo para o computador.

Com isso em mãos, foi possível comparar os diferentes resultados para garantir que os processos embarcados resultassem em valores satisfatoriamente semelhantes aos do ambiente de treinamento.

Amostra	**Pré-Proc**	**Rede**	**Asfixia**	**Surdez**	**Fome**	**Normal**	**Dor**
Surdez	Python	Python	1.24E-9	1	4.37E-10	3.08E-11	3.28E-11
	Python	C	1.24E-9	1	4.37E-10	3.08E-11	3.28E-11
	C	Python	1.24E-9	1	4.36E-10	3.08E-11	3.26E-11
	C	C	1.24E-9	1	4.36E-10	3.08E-11	3.25E-11
Normal	Python	Python	5.01E-2	6.07E-2	2.64E-2	8.63E-1	4.66E-5
	Python	C	5.01E-2	6.07E-2	2.64E-2	8.63E-1	4.66E-5
	C	Python	5.38E-2	4.72E-2	2.82E-2	8.71E-1	6.27E-5
	C	C	5.37E-2	4.71E-2	2.82E-2	8.71E-1	6.28E-5

Tabela 5: Comparação entre pré-processamento e propagação da rede, entre os ambientes de treinamento (Python) e no dispositivo (C).

5.9 Implementação da Peça Mecânica

A peça mecânica foi impressa em 3D usando um filamento de PLA. A primeira impressão conteve falhas por falta de suportes na impressão e problemas de encaixe nas peças. Tais problemas já foram levados em conta para fazer alterações nas tolerâncias do projeto para que possamos imprimir uma segunda versão.

Figura 41: Primeira Impressão 3D da Case

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões

O projeto visou a criação de um protótipo para o primeiro item da linha de produtos idealizada, *CryCare*, que contem a rede neural de classificação de choros embarcada. De mesma forma, em todas as etapas de engenharia foi buscada manter o padrão de modularidade, possibilitando a futura atualização para itens mais avançados.

Todos os subsistemas foram devidamente testados e validados, tanto de Software como em Hardware. Porém, ao testar a classificação no dispositivo final, notou-se que ela não estava se comportando como esperado e visto durante o período de treinamento, já que classifica de forma razoável as classes **Asfixia**, **Normal** e **surdez**, e de forma imprecisa as classes **Fome** e **Dor**. Ruídos no sistema de aquisição de áudio foram identificados e são candidatos a causas dessa falha.

6.2 Contribuições

O trabalho contribuiu ativamente para o estudo de classificação de áudio de recém-nascidos, visando um maior desempenho computacional para aplicação prática do modelo.

Foi criado um modelo de rede neural próprio para atingir tais requisitos, e treinados conforme um banco de dados pré existente, Baby Chillanto.

Também para atingir os requisitos, foi projetado um hardware específico para essa aplicação, que aumenta a viabilidade comercial e produtiva do projeto.

6.3 Perspectivas de Continuidade

De mais imediato, a próxima etapa de desenvolvimento do projeto é a transposição correta do treinamento do modelo de IA para o dispositivo. Para isso, deve-se revisar

o projeto de hardware para corrigir possíveis ruídos no sistema de aquisição de som e, em seguida, identificar possíveis particularidades nas amostras coletadas, para que seja possível aprimorar o treinamento da rede neural.

Além disso, o requisito de modularidade cumprido ao longo do projeto facilitará os futuros desenvolvimentos e produções dos outros modelos da linha de produtos idealizada. O modelo seguinte da linha, que introduz a conectividade via rede local, prevê o uso de uma placa ESP-32. Sua conexão já é prevista na PCB, assim como seu espaço no case mecânico e sua task prevista na implementação de firmware. E enfim, para a versão mais avançada, basta a adição de um furo na peça frontal do case, e mudanças no firmware da ESP-32.

Em paralelo, um grande passo para o aprimoramento da classificação será a obtenção de mais dados para treinamento. Para isso, buscaremos acesso a outros bancos de dados utilizados na literatura, tal como analisaremos a viabilidade da criação e classificação de dados próprios.

APÊNDICE A – PESQUISA DE MERCADO

A.1 Gráficos

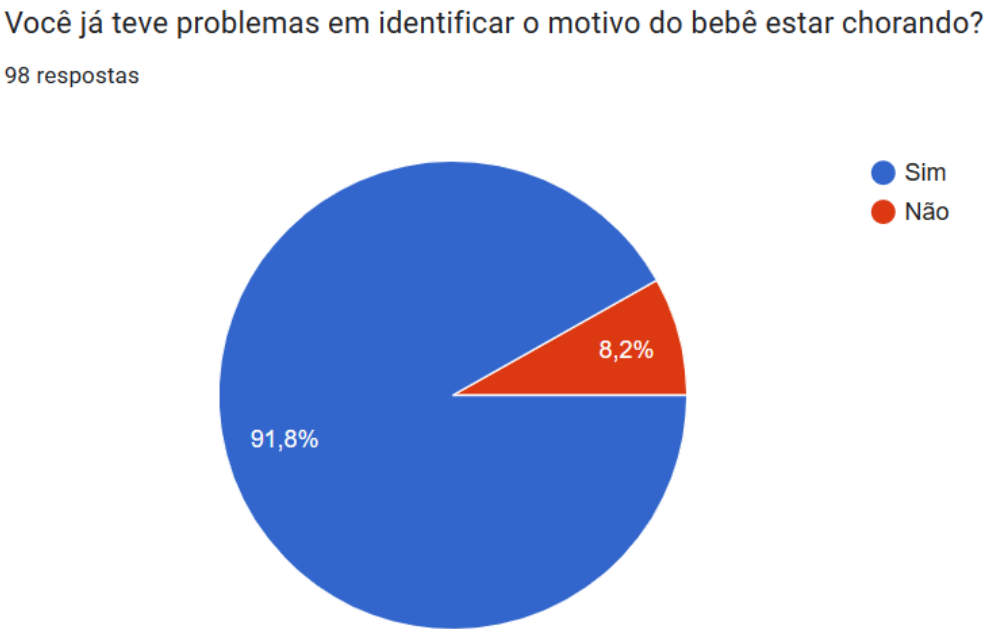

Figura 42: Circuito Amplificador

Figura 43: Circuito Amplificador

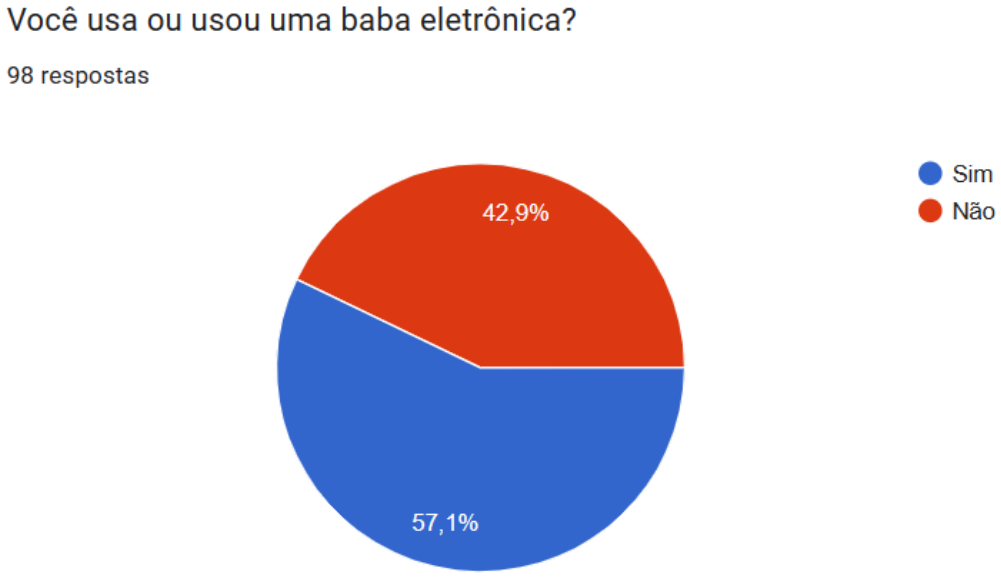

Figura 44: Circuito Amplificador

REFERÊNCIAS

BğNICğ, I.-A.; CUCU, H.; BUZO, A.; BURILEANU, D.; BURILEANU, C. Baby cry recognition in real-world conditions. In: **2016 39th International Conference on Telecommunications and Signal Processing (TSP)**. [S.l.: s.n.], 2016. p. 315–318.

FRANTI, E.; ISPAS, I.; DASCALU, M. Testing the universal baby language hypothesis - automatic infant speech recognition with cnns. In: **2018 41st International Conference on Telecommunications and Signal Processing (TSP)**. [S.l.: s.n.], 2018. p. 1–4.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

JI, C.; MUDIYANSELAGE, T. B.; GAO, Y.; PAN, Y. A review of infant cry analysis and classification. **EURASIP Journal on Audio, Speech, and Music Processing**, v. 2021, n. 1, p. 8, Feb 2021. ISSN 1687-4722. Disponível em: <https://doi.org/10.1186/s13636-021-00197-5>.

KACHHI, A.; CHATURVEDI, S.; PATIL, H. A.; SINGH, D. K. Data augmentation for infant cry classification. In: **2022 13th International Symposium on Chinese Spoken Language Processing (ISCSLP)**. [S.l.: s.n.], 2022. p. 433–437.

LAVNER, Y.; COHEN, R.; RUINSKIY, D.; IJZERMAN, H. Baby cry detection in domestic environment using deep learning. In: **2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)**. [S.l.: s.n.], 2016. p. 1–5.

MUKHOPADHYAY, J.; SAHA, B.; MAJUMDAR, B.; MAJUMDAR, A. K.; GORAIN, S.; ARYA, B. K.; BHATTACHARYA, S. D.; SINGH, A. An evaluation of human perception for neonatal cry using a database of cry and underlying cause. In: **2013 Indian Conference on Medical Informatics and Telemedicine (ICMIT)**. [S.l.: s.n.], 2013. p. 64–67.

REYES-GALAVIZ, O. F.; TIRADO, E. A.; REYES-GARCIA, C. A. Classification of infant crying to identify pathologies in recently born babies with anfis. In: MIESENBERGER, K.; KLAUS, J.; ZAGLER, W. L.; BURGER, D. (Ed.). **Computers Helping People with Special Needs**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 408–415. ISBN 978-3-540-27817-7.

RUMELHART, D. E.; MCCLELLAND, J. L. Learning internal representations by error propagation. In: _____. **Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations**. [S.l.: s.n.], 1987. p. 318–362.

RUSU, M. S.; DIACONESCU, S.; SARDESCU, G.; BRăTILă, E. Database and system design for data collection of crying related to infant's needs and diseases. In: **2015 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)**. [S.l.: s.n.], 2015. p. 1–6.