

RELATÓRIO DO TRABALHO FINAL – DETECTOR DE RONCOS

1. INTRODUÇÃO

O ronco é um fenômeno prevalente, que tanto pode ser benigno como também pode ser um sintoma de apneia obstrutiva do sono (AOS) (Xie et al, 2021). Sendo assim uma análise da qualidade do sono de uma pessoa pode ajudar na triagem e diagnostico prévio da AOS.

2. OBJETIVO

O objetivo desse projeto é detectar a presença de ronco por meio de um dispositivo embarcado.

3. HARDWARE UTILIZADO

Para a execução desse projeto foram utilizados os seguintes componentes:

- 1 Arduino Nano 33 BLE Sense + Tiny Machine Learning Shield
- Microfone MP34DT05-A incluso na placa do Arduino;

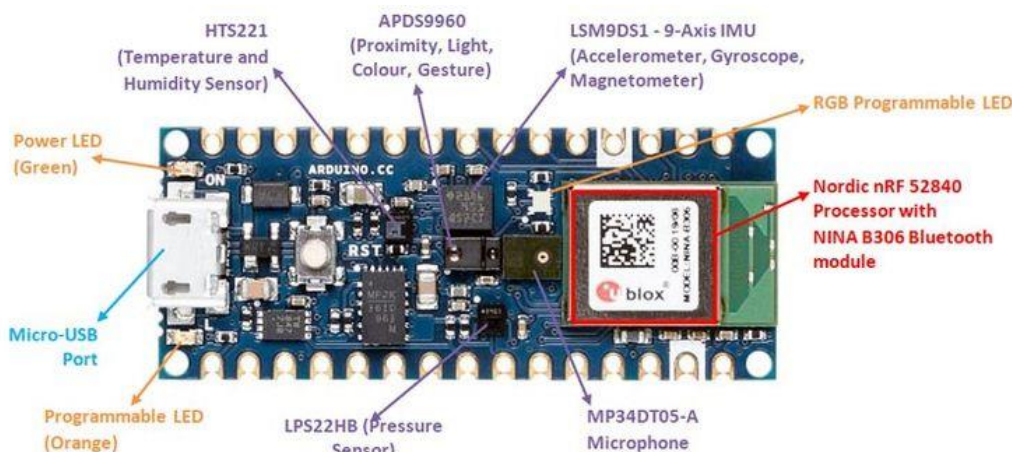


Figura 01 – Detalhes da placa Arduino Nano 33 BLE Sense.

4. DATASET

O *dataset* utilizado pode ser encontrado em <https://www.kaggle.com/datasets/tareqkhanemu/snoring>, sendo formado por um conjunto de sons de ronco e não-ronco coletados de fontes online. Cada amostra possui um segundo de duração, com uma frequência de aquisição variando entre 44,1 e 48 kHz e foram coletados sons de ronco de homens, mulheres e crianças. É importante ressaltar que a frequência de aquisição das amostras utilizadas precisou ser reduzida para 16 kHz por conta de uma limitação física do módulo Arduino e o seu microfone.

No total foram usadas 500 amostras de cada classe. Na classe de ronco 363 amostras eram somente de sons de ronco e 137 de sons de ronco com algum outro som de fundo. A classe não-ronco é composta por 50 amostras de 10 categorias diferentes. São elas:

- Choro de bebê;
- Tic-tac de relógio;
- Porta abrindo e fechando;
- Silêncio total com um leve ruído de motor;
- Descarga do vaso sanitário;
- Sirenes de veículos de emergência;
- Chuva e tempestades;
- Sons de carros de rua;
- Pessoas conversando;
- Sons de fundo de telejornais.

5. PRÉ-PROCESSAMENTO

Foram montados modelos com 3 diferentes tipos de blocos de pré-processamento, para averiguar a eficácia de cada um deles no auxílio da resolução do problema da detecção do ronco. São eles: Espectrograma, *Mel-Filterbank Energy* (MFE), *Mel Frequency Cepstral Coefficients* (MFCC).

5.1. Espectrograma

O bloco de processamento Espectrograma extrai características de tempo e frequência do sinal de entrada e é recomendado para reconhecimento de sons diferentes de fala humana ou para dados contínuos de sensores. Ele usa a transformada rápida de Fourier (FFT) para computar o espectro de frequência do sinal.

Os parâmetros desse bloco de processamento foram mantidos de acordo com a recomendação da Edge Impulse. São eles:

Tabela 1: Parâmetros para Espectrograma.

Parâmetro	Valor
Frame lenght	0,02
Frame stride	0,01
Frequency bands	128
Noise Floor (dB)	-52

5.2. MFE

O MFE também trabalha com extração de características relativas a tempo e frequência do sinal de entrada, porém ele faz uso de uma escala chamada Mel-Scale. Dessa forma, esse bloco de processamento é mais recomendado para sinais diferentes da fala humana, porém que são reconhecíveis pelo ouvido humano. Sua principal vantagem em relação ao espectrograma está na redução de entradas necessárias para a rede, deixando o modelo menor e mais rápido.

Os parâmetros configuráveis para esse bloco são chamados de *Mel-filterbanks energy features* e, assim como o espectrograma, foram mantidos de acordo com a recomendação do Edge Impulse. São eles:

Tabela 2: Parâmetros para MFE

Parâmetro	Valor
Frame lenght	0,02
Frame stride	0,01
Filter number	40
FFT length	256
Low Frequency	300
High Frequency	-
Noise floor	-52

5.3. MFCC

O bloco de processamento MFCC é bastante similar ao MFE, porém ele acrescenta um passo a mais que é uma representação mais comprimida dos *filterbanks*. Uma transformada discreta de Cosine é aplicada em cada *filterbank* para extrair coeficientes cepstrais.

Esse bloco de processamento é mais recomendado para reconhecimento de fala, mas pode ter um bom desempenho em alguns casos de sons não-humanos. A vantagem desse bloco de processamento é que ele consegue reduzir ainda mais o número de entradas do modelo, deixando o modelo final ainda menor e mais rápido.

Novamente, assim como nos casos dos blocos anteriores, os parâmetros configuráveis foram deixados de acordo com as recomendações do Edge Impulse. São eles:

Tabela 3: Parâmetros para MFCC

Parâmetro	Valor
Number of Coefficients	13
Frame lenght	0,02
Frame stride	0,02
Filter number	32
FFT length	256
Normalization window size	101
Low Frequency	300
High Frequency	-
Coefficient pré-emphasis	0,98

6. ARQUITETURA DA REDE NEURAL

Foi utilizada a arquitetura recomendada pelo Edge Impulse para os três modelos testados. Ela consiste em uma camada de entrada, uma camada de *Reshape*, uma camada de convolução com 8 neurônios em uma dimensão, uma de *Dropout* usada para prevenir *Overfitting*, mais uma camada de convolução, dessa vez com 16 neurônios, mais uma cama de *Dropout* e por fim uma camada de *Flatten* para reduzir a saída da rede para um vetor de uma dimensão. As arquiteturas de cada modelo podem ser vistas na Figura 2 a seguir. A diferença entre as arquiteturas consiste no número de atributos da camada de entrada e consequentemente no valor do *Reshape*.

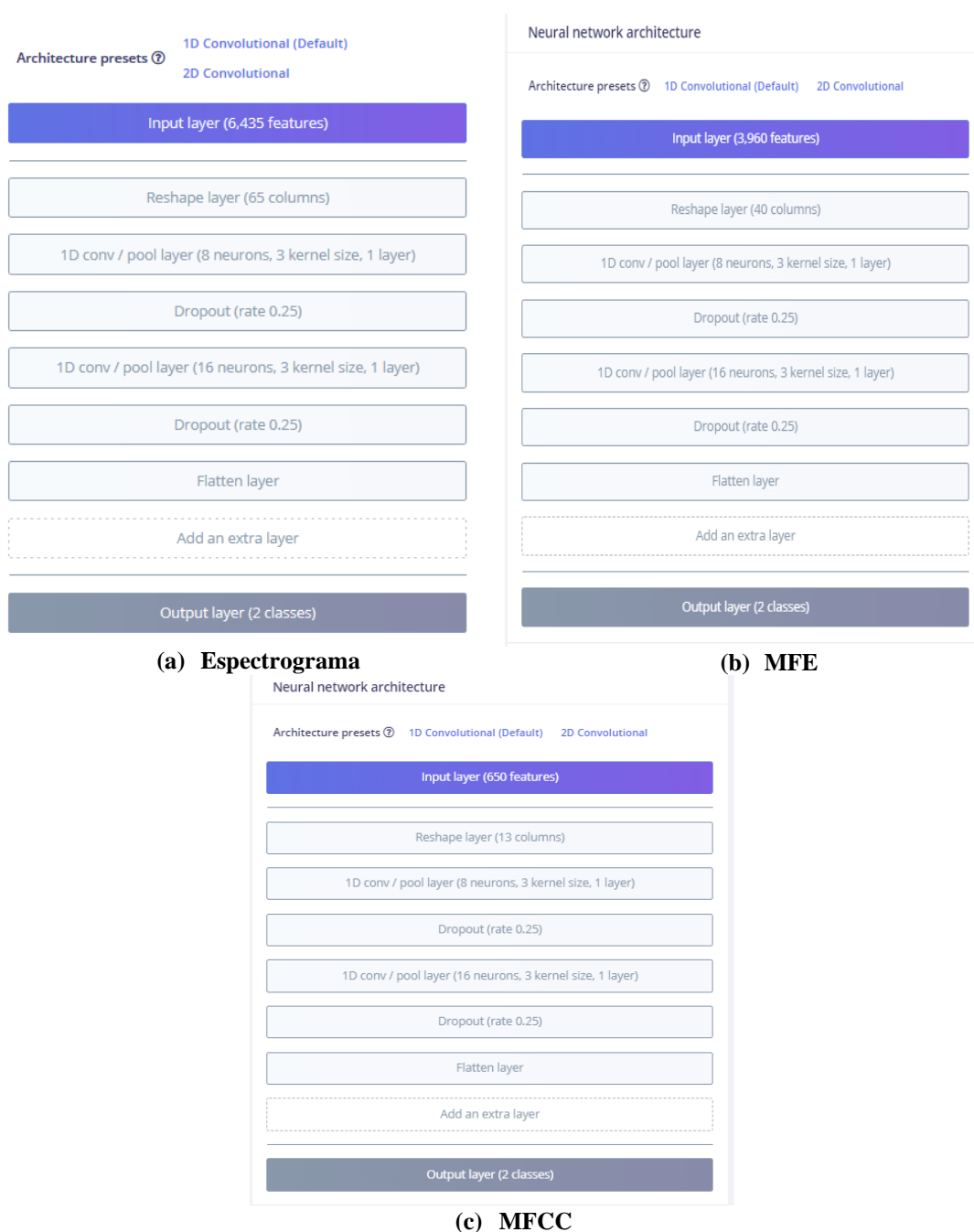


Figura 02 – Arquiteturas dos modelos utilizados.

7. TREINAMENTO

Para a avaliação das etapas de treinamento e testes dos modelos usados para esse projeto, foi usada a acurácia. E a função de perda foi mantida a *Categorical Crossentropy*, padrão do Edge Impulse.

7.1. Data Augmentation

O primeiro parâmetro avaliado na etapa de treinamento dos três modelos foi o uso ou não do *Data Augmentation*. Para isso foram realizados diversos testes de desempenho do treinamento dos modelos usando *Data Augmentation* com diferentes configurações e não usando de forma alguma.

Em todos os modelos testados, o *Data Augmentation* reduziu a taxa de acerto durante o treinamento e testes. Portanto essa ferramenta foi descartada para todos os modelos.

7.2. Número de Épocas

Para estimar o número de épocas ideal para cada modelo foram feitos testes usando um treinamento de 50 épocas para os três modelos. Em seguida foi calculada uma média da acurácia em relação a cada época para formar o gráfico mostrado na Figura X.

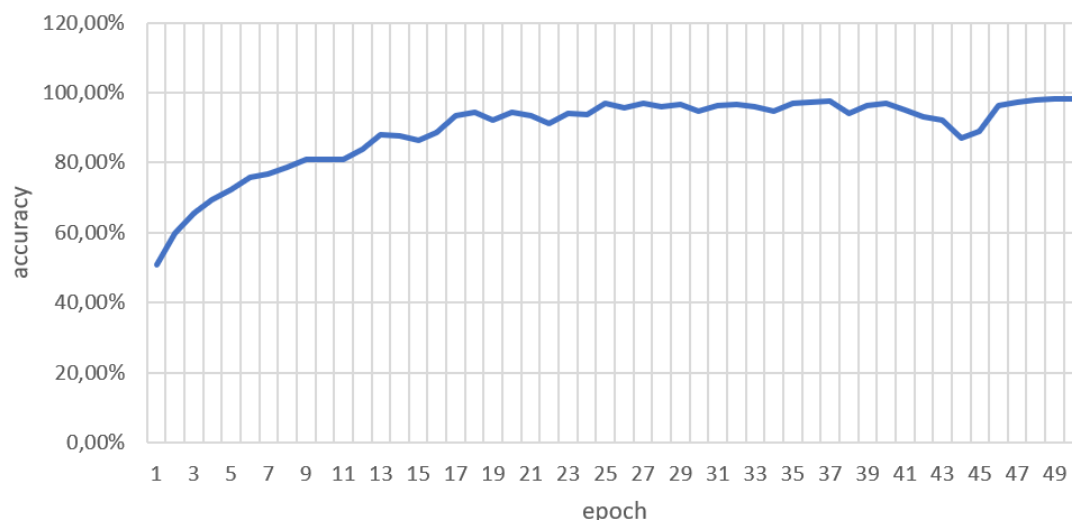



Figura 03 - Gráfico da acurácia média vs. Número de épocas. Fonte: Autor.

A partir do gráfico da Figura 03, é possível perceber que não há ganho significativo na acurácia a partir da época 25. Portanto esse foi o número de épocas usado para o projeto.

8. TESTES E DEPLOYMENT

Na etapa de testes foi realizada uma comparação entre os três modelos. Nessa comparação, vista na Figura 04, pode-se notar que o modelo com espectrograma obteve os valores mais altos de acurácia, seguido pelo MFE e por fim o MFCC, que frente aos demais modelos, não obteve bons resultados teóricos.



 ACCURACY
97.51%

	NORMAL	SNORING	UNCERTAIN
NORMAL	98.0%	1.0%	1.0%
SNORING	2.9%	97.1%	0%
F1 SCORE	0.97	0.98	

(a)

 ACCURACY
93.03%

	NORMAL	SNORING	UNCERTAIN
NORMAL	95.9%	0%	4.1%
SNORING	8.7%	90.3%	1.0%
F1 SCORE	0.94	0.95	

(b)

 ACCURACY
80.10%

	NORMAL	SNORING	UNCERTAIN
NORMAL	83.7%	6.1%	10.2%
SNORING	17.5%	76.7%	5.8%
F1 SCORE	0.83	0.84	

(c)

Figura 04 – Comparação entre os modelos com (a) Espectrograma, (b) MFE e (c) MFCC. Fonte: Autor.

Com relação aos parâmetros de *Deployment*, a comparação entre os três modelos pode ser vista na Figura 05. Pode-se perceber maior diferença em relação ao uso da RAM e da latência dos modelos. Como já era esperado, o modelo com espectrograma foi quem ocupou o maior espaço na RAM, 16,9 kB, seguido do MFE, com 11,8 KB e pelo MFCC, com 5,0 KB. O que já era esperado já que umas das vantagens do uso do MFE e MFCC era na diminuição do tamanho do modelo. Da mesma forma, o modelo com maior latência foi o com espectrograma com 356 ms, seguido do MFE com 29 ms e MFCC com 16 ms, novamente valores já esperados, visto que um modelo com menos entradas tende a ser mais rápido.

Available optimizations for NN Classifier

Quantized (int8)★

Currently selected

This optimization is recommended for best performance.

RAM USAGE

16,9K

FLASH USAGE

36,0K

LATENCY

356 ms

ACCURACY

97.51%

CONFUSION MATRIX

98.0	1.0	1.0
2.9	97.1	0

Unoptimized (float32)

Click to select

RAM USAGE

53,6K

FLASH USAGE

43,2K

LATENCY

373 ms

ACCURACY

97.51%

CONFUSION MATRIX

98.0	1.0	1.0
2.9	97.1	0

(a)

Available optimizations for NN Classifier

Quantized (int8)★

Currently selected

This optimization is recommended for best performance.

RAM USAGE

11,8K

FLASH USAGE

35,4K

LATENCY

29 ms

ACCURACY

92.54%

CONFUSION MATRIX

96.9	2.0	1.0
4.9	88.3	6.8

Unoptimized (float32)

Click to select

RAM USAGE

34,2K

FLASH USAGE

40,9K

LATENCY

60 ms

ACCURACY

92.54%

CONFUSION MATRIX

96.9	2.0	1.0
4.9	88.3	6.8

(b)

Available optimizations for NN Classifier

Quantized (int8)★

Currently selected

This optimization is recommended for best performance.

RAM USAGE

5,0K

FLASH USAGE

34,4K

LATENCY

16 ms

ACCURACY

79.1%

CONFUSION MATRIX

77.6	4.1	18.4
12.6	80.6	6.8

Unoptimized (float32)

Click to select

RAM USAGE

8,4K

FLASH USAGE

36,8K

LATENCY

40 ms

ACCURACY

78.11%

CONFUSION MATRIX

77.6	10.2	12.2
12.6	78.6	8.7

(c)

Figura 05 – Comparação do *deployment* entre os modelos com (a) Espectrograma, (b) MFE e (c) MFCC.
 Fonte: Autor.

9. RESULTADOS PRÁTICOS

Foram realizados os *deployments* de todos os modelos treinados anteriormente, iniciando pelo Espectrograma, contudo para a nossa surpresa, apesar do modelo apresentar uma alta acurácia e um baixo *LOSS* durante o treinamento e testes na plataforma *EdgeImpulse*, quando ele foi integrado a aplicação na placa do Arduino Nano, a detecção de roncos apresentou muitos falsos positivos, assim classificando silêncio como ronco.

Já o modelo MFCC apresentou um desempenho melhor que o Espectrograma, contudo inferior ao MFE.

Para o *deployment* do modelo MFE na forma de *library* do Arduino, foi utilizado como base o código `nano_ble33_sense_microphone.ino`, contido nos exemplos da *library*. Tanto o código alterado quanto a *library* estão disponibilizadas no [GDrive](#).

Nas figuras abaixo, está descrito o comportamento do kit quando existe a presença de ronco, led aceso em vermelho, e quando não existe presença de ronco, led aceso em azul. Para determinar o limiar entre presença ou não de ronco, foi utilizada um multiplicador de segurança para a classe não-ronco, assim somente quando a probabilidade da classe ronco for maior que 2 vezes o valor da classe não-ronco que de fato a aplicação irá indicar que o ronco foi detectado.

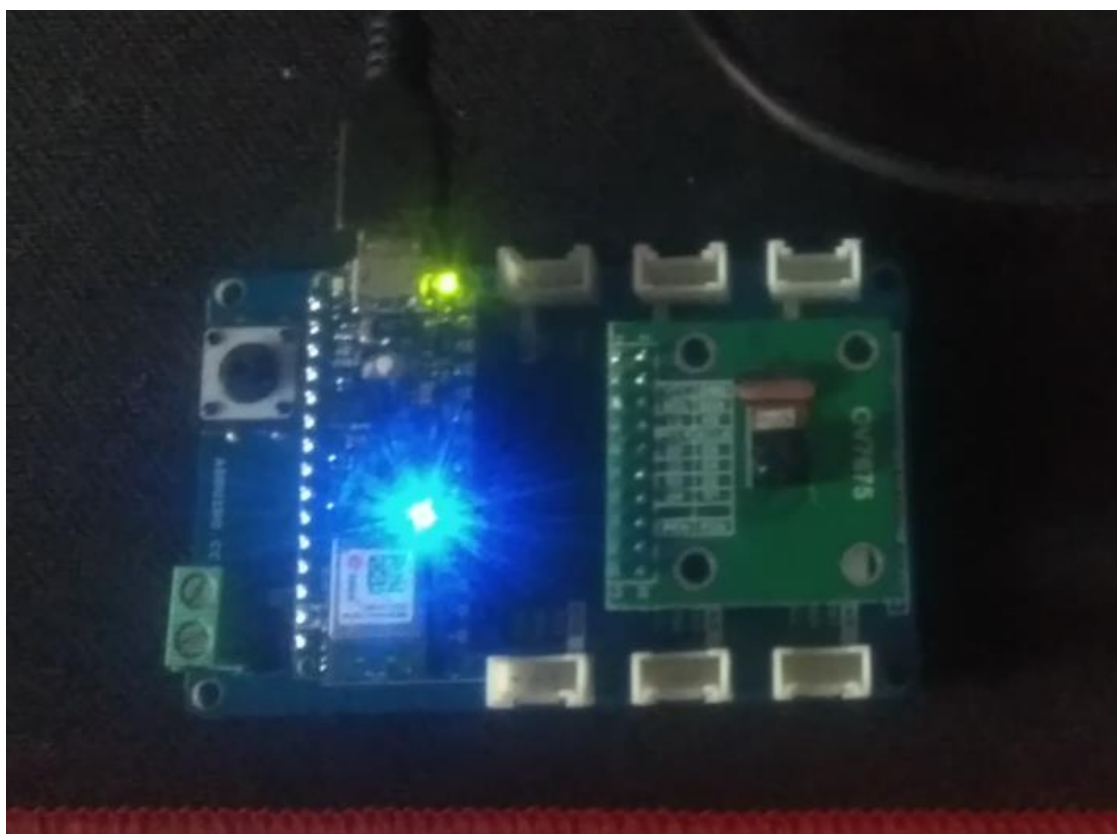


Figura 06 – Aplicação Snoring Detection com o led aceso em azul, indicando não detecção de ronco. Fonte: Autores.

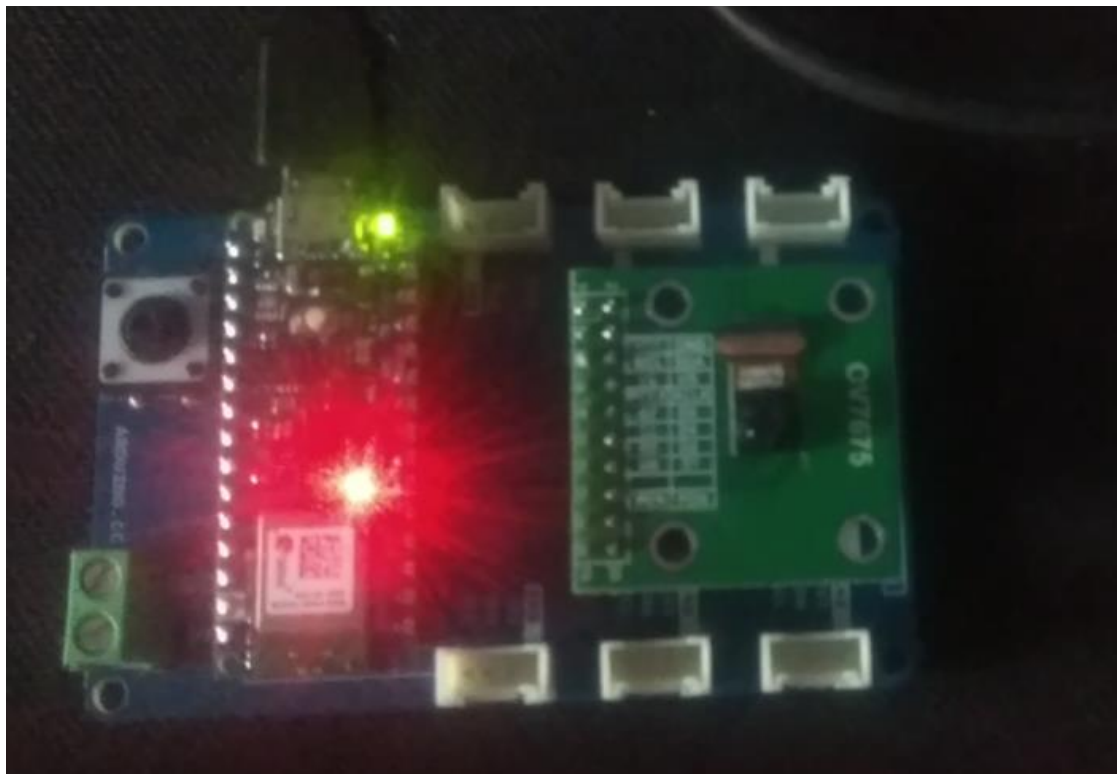


Figura 07 – Aplicação Snoring Detection com o led aceso em vermelho, indicando detecção de ronco. Fonte: Autores.

10. CONCLUSÃO

Quanto ao objetivo do trabalho, pode-se concluir que é plenamente possível realizar a detecção do ronco em um dispositivo embarcado. Também foi visto que dentre os modelos aqui testados e avaliados, o modelo com a camada de pré-processamento utilizando MFE se destacou como a melhor solução para a detecção do ronco.

Outra descoberta desse projeto foi na avaliação do modelo com espectrograma, que apresentou um ótimo desempenho teórico, com os maiores valores de acurácia nas etapas de treinamento e teste, mas que após o *deployment* apresentou um número elevado de falsos positivos, principalmente quando a entrada era de silêncio.

O modelo com MFCC obteve um resultado decepcionante, tanto na etapa de treinamento e testes quanto após o *deployment*, ficando na frente apenas do espectrograma. Entretanto, foi o modelo que apresentou menor tamanho na RAM e menor latência.