Computação em Borda para o Monitoramento de Abelhas: Detecção da Ausência da Rainha por Análise de Áudio

Jorge F. Ramos Bezerra[1], Sandy F. da Costa Bezerra[2], Ícaro de Lima Rodrigues[2], Elias Teodoro da Silva Jr.[1], Danielo G. Gomes[2], Antonio Rafael Braga[1]

[1]Programa de Pós-Graduação em Ciências da Computação (PPGCC)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

[2]Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Centro de Tecnologia – Universidade Federal do Ceará (UFC)

jorge.fernando60@aluno.ifce.edu.br, {icarodelima, sandycosta}@alu.ufc.br

elias@ifce.edu.br, {danielo, rafaelbraga}@ufc.br

Abstract. *The preservation of bees is essential for biodiversity, and the presence of the queen bee is crucial for the organization of the hive. Using cutting-edge computing, this work proposes a method to detect its absence through the analysis of hive audio in a fast, feasible and accessible way. The recordings were processed, extracting relevant acoustic features, used in the classification by Naive Bayes, KNN, MLP and Random Forest. These algorithms achieved accuracies ranging from 87.66% to 97.54%. MLP was chosen for implementation in a microcontroller, where it achieved 88.50% accuracy with an average inference time of 109 ms.*

Resumo. *A preservação das abelhas é essencial para a biodiversidade, e a presença da abelha-rainha é crucial para a organização da colmeia. Utilizando a computação em borda esse trabalho propõe um método para detectar sua ausência por meio da análise de áudios da colmeia de forma rápida, viável e acessível. As gravações foram processadas, extraindo-se características acústicas relevantes, utilizadas na classificação por Naive Bayes, KNN, MLP e Random Forest. Esses algoritmos alcançaram acurácias que variam de 87,66% até 97,54%. O MLP foi escolhido para implementação em microcontrolador, onde atingiu 88,50% de precisão com um tempo médio de inferência de 109 ms.*

1. Introdução

A sobrevivência das abelhas impacta diretamente a biodiversidade e a segurança alimentar global. Responsáveis por cerca de 90% dos serviços de polinização comercial e 35% das culturas alimentares mundiais [Kanelis et al. 2023], elas sustentam ecossistemas e cadeias produtivas essenciais. No Brasil, 85 das 141 espécies cultivadas dependem da polinização animal [Pires et al. 2016], reforçando a importância de práticas eficazes de manejo.

Diversos fatores têm contribuído para o declínio populacional das abelhas, como mudanças climáticas, doenças e uso intensivo de pesticidas [Danieli et al. 2024, Lu et al. 2024]. Além disso, a perda da abelha-rainha — responsável pela reprodução na colmeia — compromete diretamente a continuidade e a organização social de uma colônia [Santos et al. 2025]. Detectar precocemente essa ausência é essencial, mas os

métodos tradicionais baseiam-se em inspeções manuais, as quais são invasivas e normalmente imprecisas e operacionalmente limitadas.

A Apicultura de Precisão (AP), integrada à Internet das Coisas (IoT), tem se mostrado promissora para modernizar esse cenário [Braga, R. et al. 2020]. No entanto, em ambientes rurais, limitações como falta de conectividade e difícil acesso às colmeias podem comprometer a adoção prática dessas tecnologias computacionais. Alguns estudos recentes buscaram identificar a ausência da rainha com base em parâmetros como temperatura, umidade e som, aplicando algoritmos de aprendizado de máquina em computadores de alto desempenho [Otesbelgue et al. 2025]. Avanços também foram obtidos com a execução de modelos em dispositivos móveis [Santos et al. 2025], e em classificadores otimizados para resposta rápida [Rodrigues et al. 2022]. Contudo, essas abordagens ainda dependem de equipamentos relativamente caros ou processamento remoto.

Por outro lado, o paradigma da Computação em Borda pode ser uma alternativa eficaz na medida que permite o processamento de dados diretamente no local, com menor dependência de energia e conectividade [Lai et al. 2018]. Entretanto, soluções embarcadas específicas para monitorar a presença da rainha ainda são escassas, sobretudo com foco em viabilidade técnica e custo acessível.

Neste artigo, propomos um sistema de monitoramento acústico embarcado baseado em um microcontrolador ESP32, utilizando aprendizado de máquina para classificar localmente o estado da colônia. A partir da análise de áudios capturados na colmeia, características acústicas foram extraídas por Transformada de Fourier e processadas por redes neurais do tipo MLP (*Multi-Layer Perceptron*). Com isso, nossa proposta permite identificar com precisão a ausência da rainha sem necessidade de computação em nuvem, oferecendo uma alternativa prática, acessível e não-invasiva para apicultores. Nossa hipótese científica é que padrões sonoros registrados no interior da colmeia apresentam variações detectáveis, associadas à ausência da abelha-rainha, e que essas variações podem ser reconhecidas com precisão por classificadores treinados, mesmo em dispositivos com recursos computacionais limitados.

2. Materiais e Métodos

O desenvolvimento deste estudo foi estruturado em etapas sequenciais, executadas em dois ambientes computacionais complementares:

1. Ambiente de desenvolvimento e análise: computador com processador Intel Core i3 (11ª geração, 3 GHz), 16 GB de memória RAM, utilizamos a linguagem Python no ambiente Jupyter Notebook para processamento de dados e treinamento dos modelos de aprendizado de máquina;
2. Ambiente embarcado: microcontrolador ESP32 modelo WROOM-32U, com processador Xtensa Dual-Core 32-bit LX6 (240 MHz), 520 KB de RAM e 16 MB de memória Flash. Selecionamos esse microcontrolador, baseado em uma pesquisa por dispositivos que oferecessem conectividade Wi-Fi integrada, dispensando a necessidade de módulos adicionais. Entre os modelos disponíveis no mercado local, priorizou-se aquele com menor custo e que apresentasse maior frequência de clock, de memória flash e RAM. A Figura 1 apresenta o ESP32 utilizado. Nesse ambiente, implementamos o tratamento, a extração de características e a classificação dos áudios, em linguagem C, por meio da plataforma Arduino IDE.

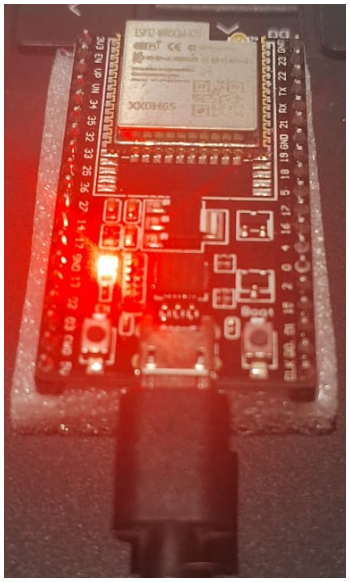

Figura 1. ESP32 WROOM-32U.

As etapas de desenvolvimento, os resultados obtidos em cada etapa e em qual ambiente as etapas foram executadas, são ilustradas na Figura 2.

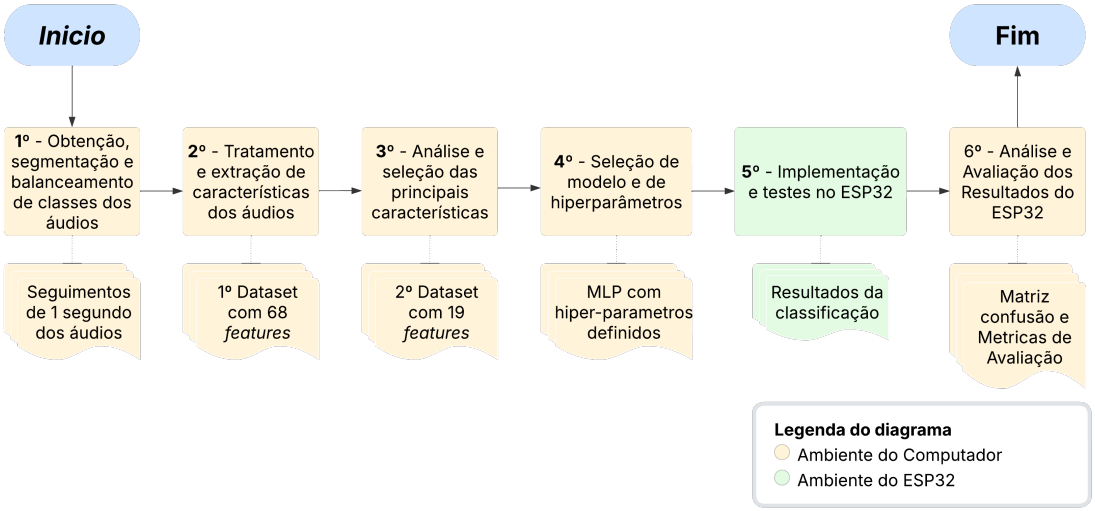

Figura 2. Fluxograma metodológico.

2.1. Obtenção, segmentação e balanceamento de classes dos áudios

Os áudios utilizados neste artigo provêm de 5 arquivos de áudio de aproximadamente 4 horas cada [Rodrigues et al. 2024]. Destes, 4 são com abelha-rainha e 1 sem abelha-rainha. Dessa forma, todos possuem um rótulo que permite identificar a presença ou ausência da abelha-rainha.

Os áudios adquiridos foram segmentados em janelas de 1 segundo e cada segmento foi convertido para o formato WAV, utilizando codificação PCM de 16 bits, canal mono e taxa de amostragem de 16 kHz. No total, foram obtidos 13974 segmentos de

áudios de 1 segundo, identificados devido ao rótulo original e alocados em uma pasta identificada como ”sem abelha-rainha”, além destes, foram obtidos 66015 segmentos de áudios de 1 segundo identificados devido aos rótulos originais. Neste último conjunto foi aplicado a técnica *Random Undersampling* (redução da classe majoritária) que remove aleatoriamente exemplos da classe majoritária para balancear as proporções, selecionando 13974 segmentos sem repetições, para armazenar em uma pasta identificada como ”com abelha-rainha”.

2.2. Tratamento e extração de características dos segmentos de áudios

Para tratar os segmentos de áudio aplicamos a Transformada de Fourier (*Fourier Transform* (FT)) [Schwartzbach 2015] e os Coeficientes Cepstrais de Frequência Mel (*Mel-Frequency Cepstral Coefficients* (MFCC)) [Abdul and Al-Talabani 2022], que são técnicas amplamente utilizadas para esta finalidade. Após esse tratamento, utilizamos as funções do módulo `ShortTermFeatures` da biblioteca `pyAudioAnalysis` [Giannakopoulos 2015] para a extração de características de curto prazo (*short-term features*) dos sinais de áudio. Com isso, foi extraído um conjunto de 68 características relevantes (*features*), gerando um conjunto de dados (*dataset*). Nesta tarefa foram usados 20 mil segmentos de áudios, sendo 10 mil com abelha-rainha e 10 mil sem abelha-rainha, escolhidos aleatoriamente e sem repetição, de suas respectivas pastas.

2.3. Análise e seleção das principais características

Nesta etapa utilizamos a técnica de Análise de Componentes Principais (PCA, *Principal Component Analysis*) [Shlens 2014] como método estatístico para redução de dimensionalidade dos dados e preservando a sua máxima variabilidade. Sendo implementado em Python com auxilio da biblioteca `scikit-learn` [Pedregosa et al. 2011]. O que permitiu a redução de 68 para 19 componentes ou características, sendo elas:

- MFCC 1 a MFCC 13, representando a envoltória espectral do sinal.
- *Energy*, energia total do sinal, indicando a intensidade do som.
- *Entropy*, entropia do sinal, medida da dispersão da energia no tempo.
- *Centroid*, centróide espectral, frequência média ponderada do espectro.
- *Bandwidth*, largura de banda espectral, dispersão das frequências ao redor do Centróide.
- *Rolloff*, frequência de *roll-off*, ponto onde uma porcentagem específica da energia espectral está contida.
- RMS (*Root Mean Square*), valor quadrático médio, potência do sinal.

A Figura 3 ilustra a relação entre os componentes principais e suas respectivas variâncias. A Figura 4 mostra a projeção dos dados nos dois primeiros componentes principais, permitindo uma visualização 2D da distribuição das classes. A Figura 5 estende essa análise ao incluir o terceiro componente principal, oferecendo uma perspectiva tridimensional para evidenciar padrões não perceptíveis em 2D. Caso as classes ”com abelha-rainha”e ”sem abelha-rainha”estejam concentradas em regiões distintas, isso indica que o PCA foi eficaz na redução da dimensionalidade e na separação das classes.

Então, implementamos a FFT e a extração dos principais componentes em Python para gerar um novo *dataset* de dimensionalidade reduzida, com os 20 mil segmentos previamente selecionados. Essa implementação ocorreu sem o auxílio de bibliotecas para evitar divergências significativas quando os dados fossem processados posteriormente em linguagem C no ESP32.

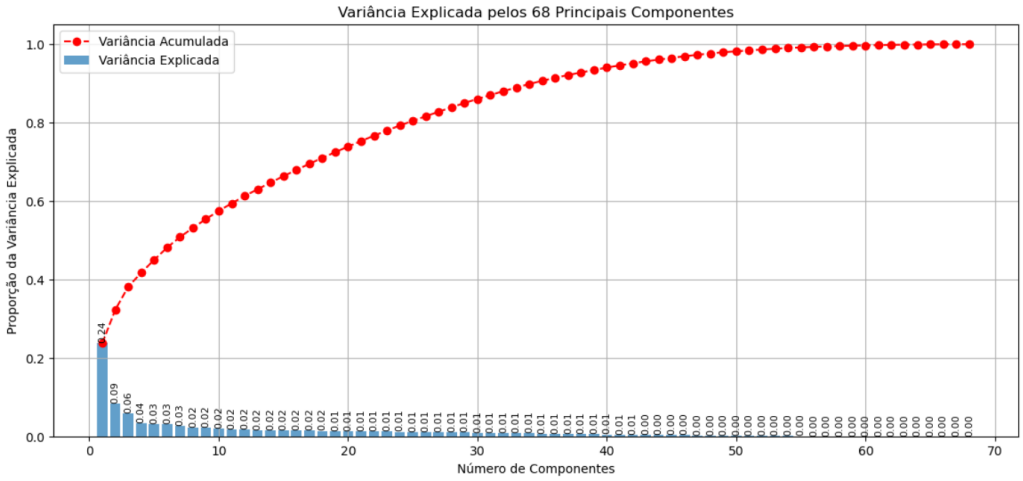

Figura 3. Variância Explicada Acumulada

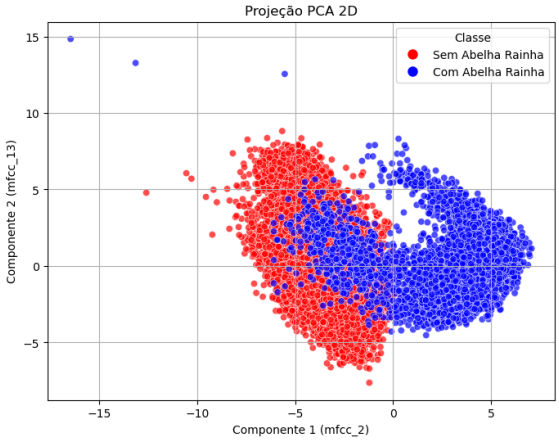

Figura 4. Os Dois Principais Componentes pelo PCA

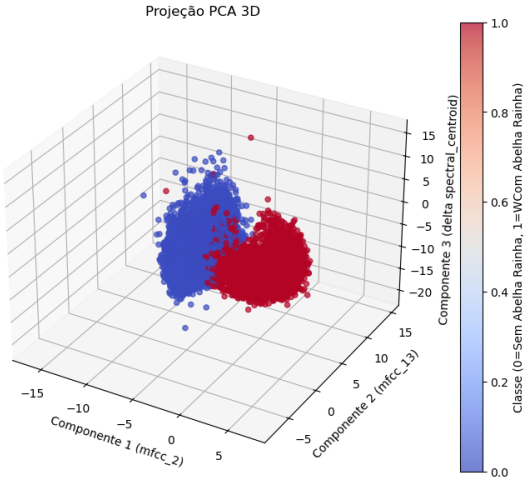

Figura 5. Os Três Principais Componentes pelo PCA

2.4. Seleção de modelo e de hiperparâmetros

Nesta etapa, implementamos quatro classificadores em Python, utilizando a biblioteca Scikit-Learn [Pedregosa et al. 2011]. Os avaliamos considerando duas métricas principais: tempo de execução, como indicador de custo computacional, e acurácia, como medida de desempenho dos modelos. Para, então, selecionar o modelo que seria implementado no ESP32. Em alguns casos, o custo computacional aumentava sem melhorias significativas na acurácia.

Os classificadores que utilizamos foram: *Naive Bayes Gaussiano* (Gaussian NB), *K-Nearest Neighbors* (KNN), *Multi-Layer Perceptron* (MLP) e *Random Forest* (RF). O Gaussian NB implementamos uma única vez, por não requerer ajuste de hiperparâmetros. O KNN implementamos quatro vezes, com diferentes valores para o número de vizinhos (*neighbors*). O MLP foram nove vezes, com variações na quantidade de neurônios das duas camadas ocultas. A camada de entrada foi mantida com vinte neurônios, correspondente ao número de características extraídas por segmento, e a camada de saída com dois neurônios, representando as classes possíveis. O Random Forest também foram nove

vezes, com diferentes combinações do número de estimadores (*estimators*) e valores do *random state*. Para aferirmos os resultados dos classificadores, utilizamos a validação cruzada estratificada de dez dobras (10-*fold cross-validation*) e o intervalo de confiança (95%).

2.5. Implementação e testes no ESP32

Na implementação no ESP32, utilizamos o plugin SPIFFS que permite carregar os arquivos na memória flash. Foram carregados 10 segmentos de áudios por rodada de teste, sendo aproximadamente 1,37 MB, no total. Realizamos 100 rodadas de testes, com 1000 segmentos de áudio, selecionados aleatoriamente e sem repetição das pastas com abelha-rainha e sem abelha-rainha. Esses segmentos não fizeram parte da elaboração dos datasets utilizados no treinamento e teste. Além disso, elaboramos um arquivo de referência contendo a identificação de cada segmento e a respectiva pasta de origem.

Para a classificação dos áudios na ESP32, foi utilizado um arquivo gerado a partir do treinamento do MLP no computador. Esse arquivo armazena os pesos calculados para cada neurônio da rede neural e é empregado na classificação dos áudios carregados na memória flash, após a aplicação da FFT e a extração de características.

Após cada rodada, os resultados foram exibidos no monitor serial do Arduino IDE e salvos em um documento de texto. Os resultados continham as seguintes informações: identificação do segmento, tempo gasto na aplicação do FFT, tempo gasto calculando as características, tempo gasto classificando, tempo total processando o segmento e o resultado da classificação (com abelha-rainha ou sem abelha-rainha).

2.6. Análise e Avaliação dos Resultados do ESP32

Nesta etapa, analisamos os resultados obtidos na classificação realizada no ESP32 e os comparamos com o arquivo de referência preparado durante a seleção dos segmentos utilizados. A partir dessa comparação, calculamos as seguintes métricas de desempenho: acurácia, recall, F1-score e F_β -score ($\beta = 0,5$), acompanhadas de seus respectivos intervalos de confiança. Além disso, foi gerada uma matriz de confusão.

3. Resultados

Nesta seção, apresentamos os resultados obtidos a partir da aplicação de diferentes técnicas e classificadores sobre os dados extraídos de áudios de uma colmeia de abelhas.

3.1. Resultados dos Testes dos Classificadores no Computador

Os classificadores seguiram a mesma proporção de treino e teste, respectivamente 80% e 20%. Seus resultados, apresentados na Tabela 1, indicam que o modelo RF obteve o melhor desempenho na classificação, mas com um custo computacional bem superior ao MLP, que apresentou o segundo melhor desempenho, com uma acurácia muito próxima ao primeiro. Por outro lado, o NB e o KNN apresentaram acurácia inferior, sendo que este último teve o maior custo computacional. O MLP (64x20) foi escolhido para implementação no ESP32.

Quanto às métricas de avaliação dos classificadores, presentes na Tabela 1, o Tempo Médio por Segmento (TMS), é o tempo da inferência de um segmento e está

apresentado em milissegundos. As métricas de Acurácia, Recall e F1-score são acompanhadas de seus respectivos Desvios Padrão (DP). Já os classificadores são apresentados junto dos hiperparâmetros testados.

Tabela 1. Comparação dos Resultados dos Classificadores

Classificador	Acurácia (DP)	Recall (DP)	F1-score (DP)	TMS (ms)
NB	0,9032 (0,0038)	0,8669 (0,0081)	0,8993 (0,0042)	0,0012
KNN (5)	0,8798 (0,0051)	0,8899 (0,0083)	0,8805 (0,0052)	0,0872
KNN (10)	0,8778 (0,0052)	0,8679 (0,0093)	0,8762 (0,0051)	0,1002
KNN (15)	0,8788 (0,0050)	0,8827 (0,0081)	0,8798 (0,0053)	0,0787
KNN (20)	0,8766 (0,0051)	0,8684 (0,0059)	0,8684 (0,0054)	0,1018
MLP (32x10)	0,9634 (0,0157)	0,9592 (0,0390)	0,9629 (0,0182)	0,0007
MLP (64x10)	0,9622 (0,0081)	0,9618 (0,0255)	0,9624 (0,0086)	0,0005
MLP (128x10)	0,9670 (0,0036)	0,9714 (0,0104)	0,9671 (0,0034)	0,0012
MLP (32x20)	0,9618 (0,0064)	0,9640 (0,0207)	0,9618 (0,0067)	0,0007
MLP (64x20)	0,9662 (0,0060)	0,9627 (0,0221)	0,9663 (0,0060)	0,0005
MLP (128x20)	0,9604 (0,0115)	0,9509 (0,0297)	0,9598 (0,0128)	0,0012
MLP (32x30)	0,9596 (0,0081)	0,9608 (0,0198)	0,9597 (0,0084)	0,0006
MLP (64x30)	0,9560 (0,0221)	0,9507 (0,0544)	0,9550 (0,0259)	0,0008
MLP (128x30)	0,9569 (0,0127)	0,9596 (0,0365)	0,9568 (0,0134)	0,0013
RF (50x21)	0,9746 (0,0026)	0,9720 (0,0041)	0,9745 (0,0026)	0,0046
RF (100x21)	0,9750 (0,0019)	0,9739 (0,0032)	0,9750 (0,0020)	0,0087
RF (200x21)	0,9750 (0,0025)	0,9733 (0,0034)	0,9749 (0,0026)	0,0168
RF (50x42)	0,9745 (0,0028)	0,9722 (0,0046)	0,9744 (0,0027)	0,0043
RF (100x42)	0,9750 (0,0018)	0,9718 (0,0041)	0,9748 (0,0019)	0,0100
RF (200x42)	0,9748 (0,0023)	0,9726 (0,0036)	0,9748 (0,0023)	0,0158
RF (50x84)	0,9741 (0,0022)	0,9707 (0,0037)	0,9741 (0,0022)	0,0045
RF (100x84)	0,9754 (0,0028)	0,9736 (0,0036)	0,9754 (0,0026)	0,0087
RF (200x84)	0,9735 (0,0018)	0,9713 (0,0039)	0,9734 (0,0019)	0,0166

3.2. Resultados do MLP 64x20

A Tabela 2 apresenta os resultados da implementação em Python, que justificam a escolha do classificador MLP, com 64 e 20 neurônios nas camadas ocultas. As métricas indicam que o modelo apresentou desempenho consistente, com altos níveis de acurácia e estabilidade, conforme observado pelos baixos desvios padrão. A aplicação da validação cruzada reforça a confiabilidade dos resultados. As demais métricas, como recall, F1-score e F_β-score, também revelam um bom equilíbrio entre sensibilidade e precisão nas predições, e os tempos de execução registrados são úteis como referência de desempenho.

3.3. Tempos de processamentos de Áudios no ESP32

Nas 100 rodadas de testes que realizamos, cada uma com 10 segmentos de áudios armazenados na memória flash do ESP32. Calculamos os tempos de execução para a aplicação da FFT, da extração de características, classificação dos segmentos e do tempo total de processamento de cada áudio. Esses tempos foram analisados, suas médias e Intervalos de Confiança (IC) foram calculados. Os resultados, em milissegundos, são apresentados na Tabela 3.

Tabela 2. Resultados para MLP 64x20

Métrica	Valor	IC	Desvio Padrão
Acurácia Média	0,9662	0,9589 – 0,9656	0,0060
Acurácia Média (Validação Cruzada)	0,9613	0,9502 – 0,9724	0,0179
Recall Médio	0,9627	0,9530 – 0,9724	0,0221
F1 Score Médio	0,9663	0,9584 – 0,9656	0,0060
F_β -score ($\beta = 0,5$) Médio	0,9619	0,9574 – 0,9664	0,0103
Tempo Total de Execução		75,18 segundos	
Tempo de Validação Cruzada		37,07 segundos	
Tempo Médio por Amostra (teste)	0,0005	0,0001 – 0,0009	0,0009

Tabela 3. Tempos de Execução e Intervalos de Confiança, no ESP32

Etapa	Tempo Médio (ms)	IC (ms)
FFT	52,01	51,91–52,11
Extração de características	16,4	16,27–16,41
Classificação	2,03	2,01–2,05
Processamento Total	109,61	108,97–110,25

3.4. Resultados da Classificação no ESP32

Os resultados obtidos foram analisados para calcular métricas de desempenho, como acurácia, intervalo de confiança da acurácia, recall e F1-score. Os resultados da classificação são ilustrados na Figura 6 e as métricas estão presentes na Tabela 4.

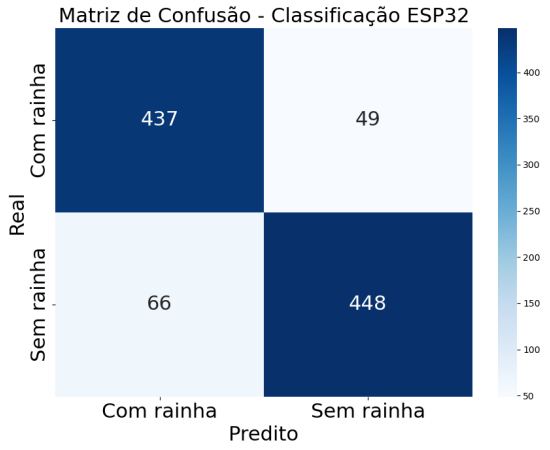

Figura 6. Matriz de Confusão.

Tabela 4. Resultados da Classificação no ESP32

Métrica	Valor
Acertos	885
Erros	115
Acurácia	0,8850
IC- Acurácia (95%)	(0,8652;0,9048)
Recall	0,8688
F1 Score	0,8837
F_β -score ($\beta = 0,5$)	0,8929

4. Conclusão

Neste artigo propomos uma solução acessível, não invasiva e de resposta rápida, com potencial para auxiliar apicultores no monitoramento automático de colmeias, especialmente em contextos com infraestrutura limitada. Mostramos a viabilidade do uso do microcontrolador ESP32 para a classificação local da ausência da abelha-rainha em colmeias com

base em características acústicas processadas por modelos de aprendizado de máquina. O sistema apresentou desempenho compatível com aplicações de computação em borda, com tempo médio de inferência de $109,61$ ms e acurácia de $0,8850$. Embora inferior à acurácia observada no ambiente de desenvolvimento ($0,9656$ em Python), essa redução não inviabiliza o uso prático, sobretudo considerando as limitações computacionais do dispositivo.

Em perspectiva, sugerimos a integração do sistema a módulos de aquisição de áudio em campo, permitindo o envio de alertas ao apicultor via conectividade Wi-Fi. Avaliações também poderão ser estendidas para microcontroladores com maior capacidade de processamento e para arquiteturas de redes neurais mais complexas. Adicionalmente, recomenda-se a inclusão de novas fontes de dados, como sinais de vibração e imagens, visando aprimorar a acurácia e robustez do sistema em ambientes reais.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Danielo G. Gomes agradece ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa de produtividade (processo 311845/2022-3).

Referências

Abdul, Z. K. and Al-Talabani, A. K. (2022). Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, 10:122136–122158.

Braga, R., A., G. Gomes, D., Rogers, R., E. Hassler, E., M. Freitas, B., and A. Cazier, J. (2020). A method for mining combined data from in-hive sensors, weather and apiary inspections to forecast the health status of honey bee colonies. *Computers and Electronics in Agriculture*, 169:105161.

Danieli, P. P., Addeo, N. F., Lazzari, F., Manganello, F., and Bovera, F. (2024). Precision beekeeping systems: State of the art, pros and cons, and their application as tools for advancing the beekeeping sector. *Animals*, 14(1).

Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. GitHub repository, https://github.com/tyiannak/pyAudioAnalysis.

Kanelis, D., Liolios, V., Papadopoulou, F., Rodopoulou, M.-A., Kampelopoulos, D., Siozios, K., and Tananaki, C. (2023). Decoding the behavior of a queenless colony using sound signals. *Biology*, 12(11).

Lai, L., Suda, N., and Chandra, V. (2018). Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*.

Lu, Y., Hong, W., Fang, Y., Wang, Y., Liu, Z., Wang, H., Lu, C., Xu, B., and Liu, S. (2024). Continuous monitoring the queen loss of honey bee colonies. *Biosystems Engineering*, 244:67–76.

Otesbelgue, A., Rodrigues, I., dos Santos, C., Gomes, D., and Blochtein, B. (2025). The missing queen: a non-invasive method to identify queenless stingless bee hives. In *Apidologie*. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). scikit-learn: Machine learning in Python.

Pires, C. S. S., Pereira, F. d. M., Lopes, M. T. d. R., Nocelli, R. C. F., Malaspina, O., Pettis, J. S., and Teixeira, E. W. (2016). Enfraquecimento e perda de colônias de abelhas no brasil: há casos de ccd? *Pesquisa Agropecuária Brasileira*, 51(5):422–442.

Rodrigues, I., Melo, D., and Gomes, D. (2024). Bioacoustic dataset of sudden queen loss in an apis mellifera l. honeybee colony. In *Anais do XV Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 215–218, Porto Alegre, RS, Brasil. SBC.

Rodrigues, I., Melo, D., Silva, D., Rybarczyk, Y., and Gomes, D. (2022). Padrões bioacústicos como identificadores precisos da presença de rainha em colmeias de abelhas melíferas. In *Anais do XIII Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 11–20, Porto Alegre, RS, Brasil. SBC.

Santos, I. R. d., Araújo, F. H. D. d., and Magalhães, D. M. V. (2025). Análise comparativa de modelos de classificação de áudio de colmeias de abelhas em dispositivos portáteis android com onnxruntime. *Brazilian Journal of Development*, 11(2):e78007.

Schwartzbach, C. (2015). A transformada de fourier e o processamento eletrônico dos sinais.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.