



RESEARCH ARTICLE

OPEN ACCESS

DEVELOPMENT OF A TINYML-BASED DEVICE FOR AUTOMATIC DETECTION OF POULTRY DISEASES USING CHICKEN VOCALIZATIONS

Minh Nguyen-Ngoc¹, Bien Nguyen-Quang², Duan Luong-Cong^{*3}

^{1,2,3}Faculty of Electronics Engineering 1 & EDA Lab, Posts and Telecommunications Institute of Technology, Vietnam.

¹<https://orcid.org/0000-0002-7446-2131>²<https://orcid.org/0009-0007-5518-1231>³<https://orcid.org/0009-0003-6207-1170>

E-mail: *minhnn@ptit.edu.vn, biennq@ptit.edu.vn, duanlc@ptit.edu.vn

ARTICLE INFO

Article History

Received on: March 04, 2026

Reviewed on: April 04, 2026

Accepted on: May 01, 2026

Published on: July 31, 2026

Keywords:

Precision farming,
Poultry disease,
Edge computing,
TinyML,
Endpoint detection.

ABSTRACT

Poultry farming provides essential protein but faces challenges from diseases that affect bird welfare and farm economics. Traditional detection methods are slow and costly, underscoring the need for real-time, low-cost solutions enabled by precision livestock farming technologies. This paper presents a TinyML-based device for the automatic detection of poultry diseases by analyzing chicken vocalizations. The proposed system uses edge computing on a microcontroller to process audio data on-site, without requiring a constant internet connection or powerful hardware. Key steps include noise reduction with FIR filters, endpoint detection via a double-threshold short-time energy method, and feature extraction using Mel-Frequency Energy. A lightweight neural network with 1D-CNN and DNN layers classifies sounds as healthy, unhealthy, or noisy. We developed a prototype on the ESP32-S3 MCU, achieving 90.58% test accuracy and 0.98 F1-score for unhealthy detection. The device processes 3-second audio in 380 ms of preprocessing and 97 ms per inference window, using only 419 KB of flash and 258 KB of RAM. This low-cost solution enables scalable IoT integration for early disease intervention in poultry farms.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

Poultry farming plays a vital role in the agricultural sector by providing an affordable source of protein for a rapidly growing global population. However, as production intensifies to meet rising demand, the quality and health of poultry become critical concerns, not only for animal welfare but also for the economic viability of poultry operations [1] [2]. Traditional diagnostic methods, such as oocyte counting in stool or intestinal scrapings, microbial isolation, and PCR, are accurate but suffer from several drawbacks: they are labor intensive, time consuming, and often costly [3] [4]. These limitations underscore the need for rapid, efficient, and cost-effective techniques for early disease detection in large-scale poultry farms. Recent advancements have steered research towards automated detection systems that leverage artificial intelligence (AI), Internet of Things (IoT), and machine learning (ML) techniques. Early studies predominantly focused on imagebased disease detection using convolutional neural networks (CNNs) to analyze visual features such as feather condition, eye clarity, and beak anomalies [5]. Additionally, behavioral monitoring through video analysis has been explored to capture abnormal movements that may indicate distress or illness [6]. While these methods provide valuable insights, they generally require substantial computational resources and often operate offline, limiting their real-time applicability in resource-constrained environments [7]. An emerging line of research has begun to explore the use of acoustic signals as a non-invasive indicator of poultry health. Changes in chicken vocalizations, such as variations in frequency, amplitude, and temporal patterns, can serve as early markers of respiratory infections and other diseases [8] [9]. Signal processing techniques, including Mel-Frequency Cepstral Coefficients (MFCCs) extraction and spectrogram analysis, have been effectively combined with both traditional ML algorithms and deep learning models (e.g., recurrent neural networks and transformers) to classify the health status of birds [5-10]. Previous studies have demonstrated promising results, with reported accuracies ranging from 83% to above 95% under controlled experimental conditions. For example, research by Sadeghi et al. [11] showed that neural network-based analysis of chicken vocalizations could distinguish between healthy birds and those infected with specific pathogens with nearperfect accuracy. Similarly, Banakar et al. [12] employed time-frequency analysis techniques combined with support vector machines (SVM) to differentiate between multiple viral infections in chickens, achieving accuracies exceeding 90% in some cases.

The rapid expansion of IoT in agriculture has enabled the development of smart farming systems that continuously monitor environmental and physiological parameters. In poultry farms, IoT devices, ranging from temperature and humidity sensors to microphones, allow for real-time data collection and analysis. However, transmitting all raw data to centralized servers can be inefficient and may introduce delays in decision making. This is where TinyML, the deployment of machine learning models on low-power microcontroller units (MCUs), becomes a game changer. By processing data on-device, TinyML reduces latency, lowers energy consumption, and minimizes dependence on high-speed network connectivity. Recent works have demonstrated the feasibility of integrating TinyML with IoT platforms to analyze acoustic data for immediate detection of stress or disease symptoms in poultry, achieving real-time performance with minimal resource consumption [6-13]. Despite these advances, many existing approaches are limited to offline analysis or require high computational power that is impractical for deployment in typical farm environments. The overlapping of vocalizations in noisy farm settings further complicates the detection process, as background noise can obscure subtle changes in the acoustic signals of individual birds. These challenges highlight the urgent need for a system that can accurately perform realtime acoustic analysis on-site using lightweight and energyefficient models. In response to these challenges, our research makes the following key contributions:

- Developing a TinyML model capable of detecting poultry diseases through acoustic signal analysis. Our model is designed to operate with low resource consumption while delivering real-time performance.
- Implementation a prototype system that integrates this TinyML model on an MCU platform, enabling ondevice data processing and continuous monitoring of poultry health.

By integrating IoT and TinyML technologies, our approach offers a scalable and efficient solution to the limitations of traditional poultry disease detection methods. This system not only facilitates early intervention to mitigate disease spread but also enhances the overall sustainability and economic efficiency of poultry farming. The remainder of this paper is organized as follows. Section 2 details the materials and methods used in this study. In Section 3, the materials and methods employed in the study are presented. Finally, Sections 4 presents the results, discussion, and conclusions of our findings, respectively.

II. MATERIALS AND METHODS

II.1 DATASET

In this study, we utilized a dataset comprising 346 audio signal files compiled by S. Adebayo et al., which were categorized into three distinct folders: healthy, noisy, and unhealthy. The dataset was procured by dividing 100-dayold poultry into separate groups, with data collection conducted thrice daily using microphone devices. Unhealthy audio segments encompassed chicken coughs, snores, and rale sounds, whereas noise segments comprised background noise and sounds associated with poultry activities. Recordings were captured with a resolution of 24 bits at a sampling rate of 96 kHz and stored in the WAV format. Additionally, they compared the power spectra of healthy and unhealthy signals, revealing a notable spike of approximately 0.2 radians in the frequency content of anomalous sounds, in contrast to the distribution observed in normal sounds around the y-axis [5]. Our research focuses on leveraging machine learning, specifically TinyML, to automatically extract features and detect diseased chickens based on sound patterns. Our primary aim was to implement the entire computational process on a microcontroller (MCU) with resource constraints. To enhance computational efficiency prior to data processing, we lowered the sampling frequency to 16 kHz, decreased the resolution to 16 bits, and quantized the data range of -1.0 to 1.0 into the range of -32,768 to 32,767. These pre-processing steps are essential for enhancing the performance of our machine learning model during processing and training. This pre-processing step is facilitated through Python utilizing librosa [14] and soundfile [15] libraries, which provide robust support for audio processing tasks.

II.2 NOISE REDUCTION AND FRAMING

Numerous studies have consistently highlighted the prevalence of chicken vocalizations within the frequency range of 1.5 kHz to 5 kHz [4] [10-16]. Additionally, analysis of the dataset compiled by Adebayo et al. underscores that the primary frequency range of poultry sounds falls between 1 kHz and 5 kHz, as illustrated in Figure 1.

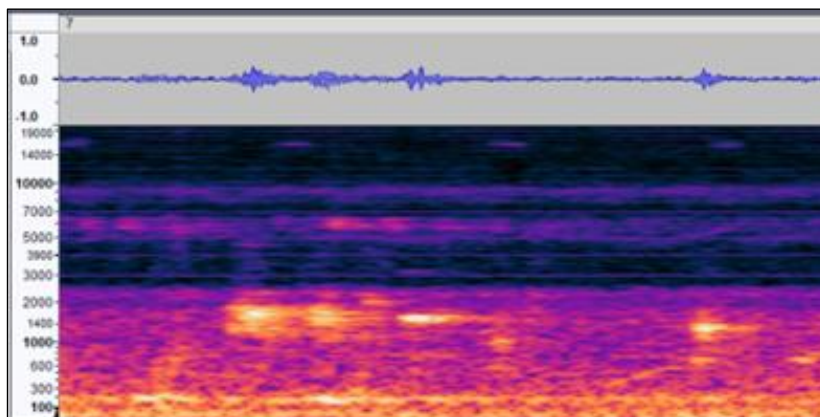


Figure 1: Spectrogram of chicken vocalization segment.

Source: Authors, (2026).

FIR filters are well known for their adaptability and precise delineation of signal characteristics within designated frequency spectra, making them particularly adept at discerning and examining poultry vocalizations within these frequency bounds. When employed in devices with restricted computational resources, such as MCUs, FIR filters have significant advantages. They ensure operational stability, manifest linear phase responses, facilitate precise frequency manipulation, and boast streamlined implementation, owing to the absence of feedback loops. These qualities underscore the suitability of FIR filters for signal processing endeavors in resource-constrained environments such as MCUs. In our study, FIR filters were leveraged for data pre-processing and subsequent implementation on the device platforms. The vocalization of chicks exhibits temporal variability, which makes direct analysis challenging. The direct analysis of audio requires significant computational resources. Nonetheless, within short durations of 10–30 ms, vocalizations tend to be relatively stable. In this study, the window function had a duration of 25 ms, corresponding to 400 sampling points. Additionally, the step length was set to half the window duration, equating to 200 sampling points (12.5 ms).

II.3 VOCALIZATION ENDPOINT DETECTION

The recorded audio contained not only chicken vocalizations but also background noise and frequent periods of silence. Achieving a more accurate analysis of chicken vocalization requires separating the chick vocalizations from the surrounding audio segments. To address these challenges, we employ a double-threshold method that incorporates two distinct features. The double-threshold technique employs both Short-Time Energy (STE) and Average Energy (AE) analyses to identify the location of the audio segments. The STE measures the energy content of sound within brief temporal windows and serves as a proxy for sound intensity. Higher STE values correlate with louder sounds, thereby facilitating the detection of sound variations, particularly in noisy environments. This enabled segmentation of the relevant audio segments for subsequent analysis. However, AE provides a measure of the overall energy level across the entire audio segment, offering a comprehensive perspective on its energy characteristics.

$$STE_k = \sum_{i=1}^{len} [y_k(i)]^2 \quad (1)$$

$$AE_j = \frac{\sum_{i=1}^{len} [y_k(i)]^2}{\Delta t} \quad (2)$$

Chick vocalizations typically have a brief duration ranging from 0.1 to 1.0 seconds [4] [16] [17]. Using this attribute, the threshold duration can be adjusted to encompass 8–80 frames for the detection of chick vocalizations. The process of identifying vocalization endpoints involves the following steps:

- 1) Compute the STE for each frame (with 400 sample points).
- 2) If the STE of a frame exceeds the threshold STE_{high} , the frame is designated as the start of a segment.
- 3) If the STE is below the threshold STE_{low} , the frame is marked as the end of the segment. Otherwise, the segment length was extended.
- 4) Once the endpoint of the segment is determined, if the segment duration falls within the range of D_1 to D_2 , the segments are merged into one. Otherwise, the segment is discarded.
- 5) The AE of the entire segment is calculated. If the AE fell below the threshold AE_{thresh} , the segment was discarded. Otherwise, proceed to the feature extraction stage with this segment.

The STE thresholds (STE_{low} and STE_{high}) were calculated using the following formula:

$$STE_{thresh} = STE_{min} + \alpha \times (\overline{STE} - STE_{min}) \quad (3)$$

where $\overline{STE}$ represents the mean of STE value across all frames, STE_{min} denotes the minimum STE value among all frames, and the parameter α is assigned values of 0.1 and 0.4 for STE_{low} and STE_{high} , respectively. The thresholds for segment duration D_1 and D_2 were set to 8 and 80 frames, respectively. The outcome of the endpoint detection process is illustrated in Figure 2, where the blue line represents the raw audio signal, the green line depicts the STE values of segments, and the red line indicates the trimmed audio segments identified by this process. Finally, AE_{thresh} was set to $3e-05$ based on our experimentation with the current dataset.

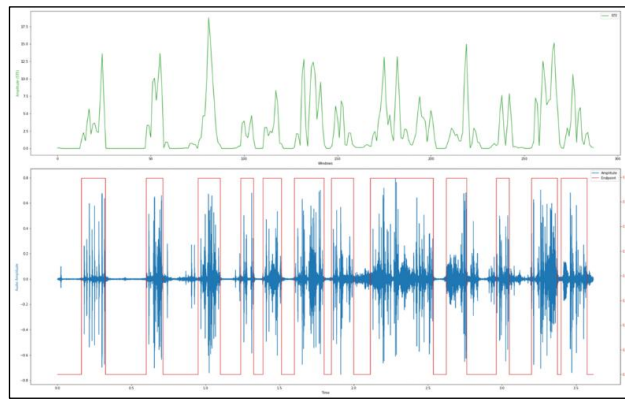


Figure 2: Vocalization endpoint detection using double threshold technique.

Source: Authors, (2026).

Upon completion of this step, the dataset comprises 8,130 audio segments, equating to approximately 41 min of audio data. Among these segments, 2,067 was categorized as belonging to the healthy group, 3,482 to the unhealthy group, and 2,581 to the noisy group. Each audio segment had a duration ranging from 100 ms to 1,000 ms.

II.4 AUDIO FEATURES

While MFCCs are widely utilized in conventional audio signal processing, log-mel spectrograms emerge as the predominant feature in deep learning, with raw waveforms or intricate spectrograms following closely. Raw waveforms sidestep manually crafted features, enabling enhanced utilization of the deep learning models' modeling process to acquire task-optimized representations [16] [18] [19]. In this study, we experimented with MFCCs and MFE for feature extraction. The results reveal that MFE exhibits slightly higher accuracy than MFCCs when running with our model design. Meanwhile, the resource usage for MFE is lower compared to using MFCC. Therefore, MFE was selected as the feature extraction step in our model. With audio windows set at 250ms, the MFE (MelFrequency Energy) block was configured with a frame length of 50ms, frame stride of 25ms, filter number of 40, FFT length of 512, and a bandpass filter (BPF) spanning from 1,000 to 5,000 Hz. The output data of the MFE block consisted of a 40×9 array for each audio window, as Figure 3. This array served as an input to the machine learning model, which will be discussed in the subsequent section.

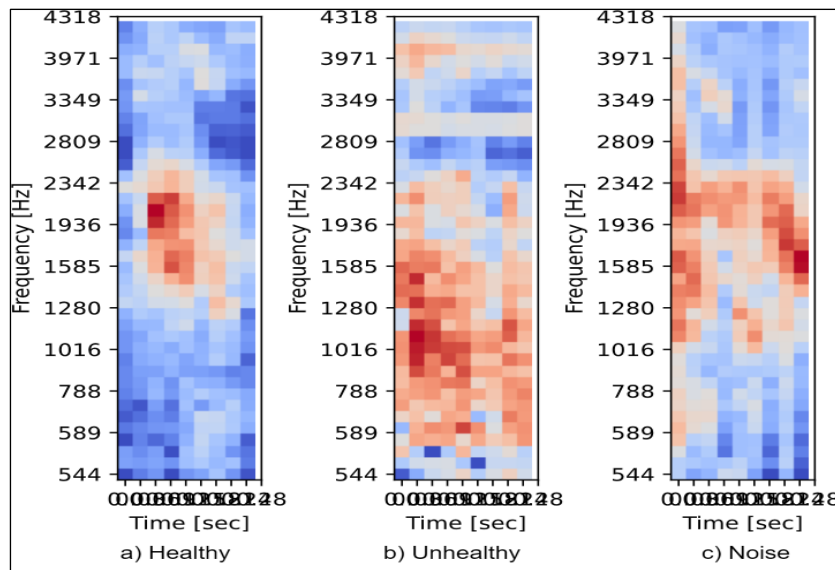


Figure 3: Output of MFE extraction for a) healthy, b) unhealthy, and c) noise audios.
Source: Authors, (2026).

II.5 TINYML MODEL DEVELOPMENT

We uploaded all segment audios and trained a model based on the Edge Impulse [20] platform to rapidly identify an appropriate model structure. After pre-processing, the dataset was split into three groups: training (60%), validation (20%), and testing (20%). Therefore, 4,878 samples were used for training, 1,626 samples for validation, and 1,626 samples for testing. Our machine learning model is initiated with two 1DCNN layers, consisting of 128 and 64 kernels. Following this, the data were flattened and passed through a twolayer DNN network with 128 neurons and 96 kernels in succession. Figure 4 shows the detailed architecture of the model network.

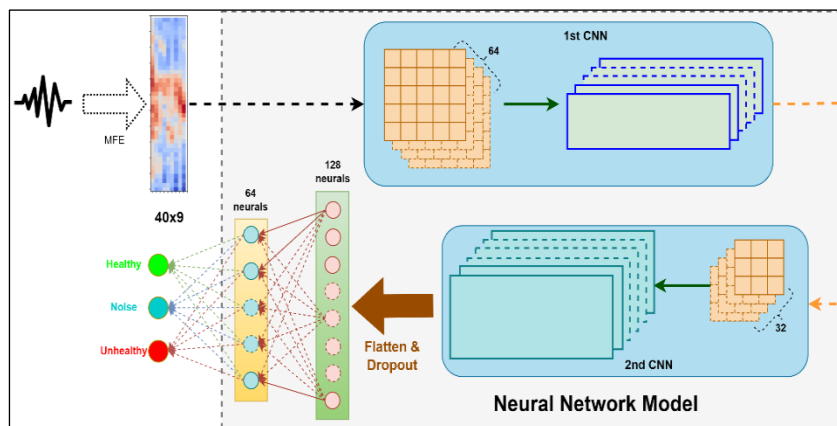


Figure 4: Neural network architecture.
Source: Authors, (2026).

III. DEVICE IMPLEMENTATION

All the steps outlined in Section II were executed using the high-level programming language Python, augmented by several efficiently supported libraries. With the aim of deploying a real-time edge device capable of audio data collection, processing, and inference, we initiated the development of a solution and embedded software program tailored for the device. The MCU selected for this study is the ESP32-S3, notable for its 384 KB ROM, 512 KB SRAM, and 4MB Flash memory capacities, along with its ability to operate at a maximum clock speed of 240 MHz. Furthermore, this MCU boasts instruction sets tailored to support TinyML and DSP algorithms, enhancing its suitability for our research purposes. A flowchart of the execution of the device is shown in Figure 5.

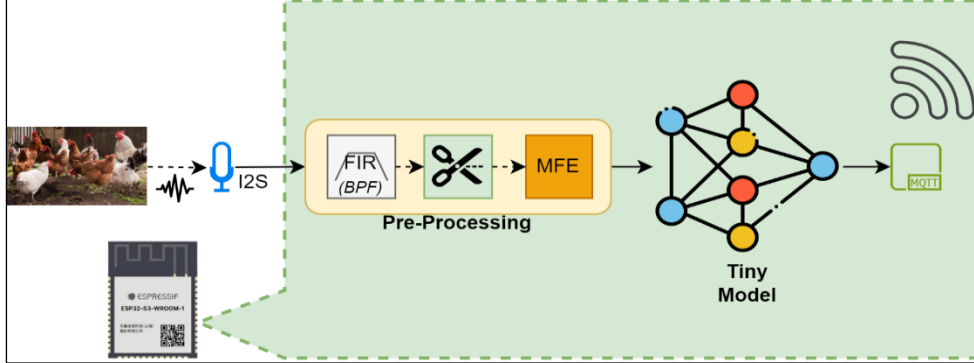


Figure 5: Implementation of device.
Source: Authors, (2026).

Figure 6 illustrates the detailed firmware architecture implemented on the device. The architecture is designed to support continuous, real-time audio acquisition and processing while maintaining low latency and efficient resource utilization. First, audio data are collected into four buffers of the MCU via Direct Memory Access (DMA) with a sample rate of 16,000 Hz and a resolution of 16 bits. Buffers can store a maximum of 3 s of audio samples. Periodically, the main program checks the audio buffers for new data. Upon the detection of new data, the data are passed through an FIR filter and subsequently saved to the ring buffer. The ring buffer consisted of three arrays: Audio Sample, STE Base, and STE window. The Audio Sample array contained signed integer (int16_t) data obtained from the FIR filter. The STE Base array computes the energy for every 200 samples, whereas the STE of the window array calculates the energy for each slice window comprising 400 samples. To optimize the computational efficiency of the MCU, we partitioned the Ring Buffer into eight equally sized segments, each capable of storing data for 0.5 seconds. For each segment, along with the three primary data arrays, we introduce two additional variables: STE_{min} , which captures the minimum STE value within the segment's audio window, and STE_{avg} , which represents the average STE value of the respective audio segments.

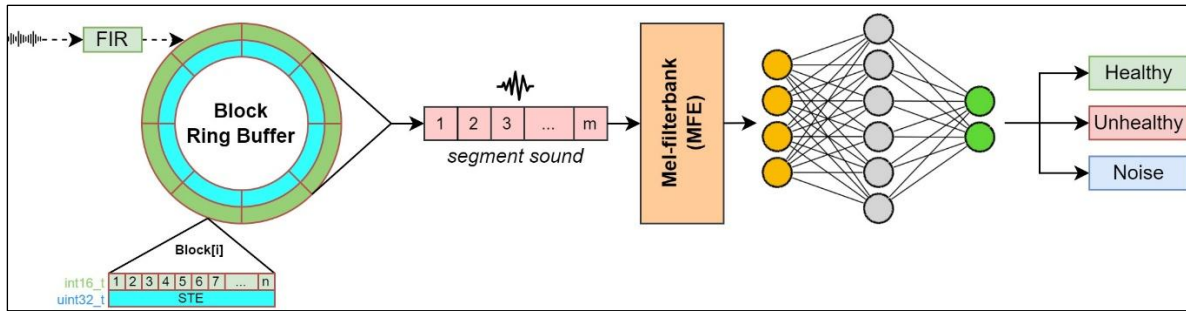


Figure 6: Firmware flow of edge device.
Source: Authors, (2026).

Instead of recalculating all the parameters with each new data point, our method efficiently computes only when the entire segment is updated. By leveraging the STE_{min} and STE_{avg} values from all segments, our program rapidly calculated STE_{low} and STE_{high} , resulting in faster computation and reduced computational overhead. These calculated thresholds facilitate the endpoint detection algorithm in identifying suitable audio segments. These segments were then split based on their length and processed through the MFE extraction step before being fed into the TinyML model. The segment sounds were split into smaller segments, each with a length of 250ms and slice window of 125ms. The number of segments to be calculated follows Equation 4, where N represents the number of segments, L denotes the length of the audio (ms), and w represents the length of the window slice, which is 125ms in this case. Each audio segment is passed through the MFE block, resulting in an output matrix with dimensions of 40×9 . This matrix is then fed into the machine-learning model, which outputs the detection probabilities for the three labels. These probability values were averaged over N audio segments to determine the identified sound types (healthy, unhealthy, or noisy). To record audio from the testing environment, we opted for the INMP441 microphone, which is capable of transmitting audio via I2S with a resolution of 24-bit and maximum sampling frequency of 48kHz. In addition, we included a TFT screen to monitor and evaluate the functionality of the devices, displaying the quantity of healthy and unhealthy chicken vocalizations during testing. All the components were integrated into a fabricated box using a 3D printer with dimensions of 87x44x26mm, as depicted in Fig.

The implementation of the embedded software for the ESP32S3 MCU was performed in accordance with the established procedure, utilizing the C++ programming language with the aid of Visual Studio Code, Platform IO, and libraries provided by Edge Impulse. The firmware is uploaded to the device via UART to create a complete device and commence the testing process.

IV. RESULTS AND DISCUSSIONS

After extensive training, validation, and testing procedures, our model demonstrated impressive performance, achieving an accuracy of 92.8% during training and 90.58% during testing using a quantization (int8_t) model. Figure 7 presents the detailed confusion matrix for the test set. Although the overall accuracy may not be exceptionally high, our model displayed remarkable proficiency in identifying diseased chickens, achieving a remarkable F1-score of 0.98. Further analysis of the dataset sources revealed that audio files labeled as "Noise" frequently contained significant chicken sounds, which could potentially influence the differentiation between the "Healthy" and "Noise" labels.

noise	65	5	759	0.89	0.91
unhealthy	9	1124	12	0.98	0.98
healthy	745	16	87	0.89	0.88
	healthy	unhealthy	noise	F1-score	Precision

Figure 7: Confusion matrix for model evaluation on the test set with 8-bit quantization.

Source: Authors, (2026).

In addition, the model underwent quantization to optimize the size and inference time. According to the report from Edge Impulse, this model uses 248 KB of flash memory, 12.4KB of RAM, and requires 92ms for porting and inference on an ESP32-S3 MCU. This model, along with the pre-processing algorithm, is implemented on the edge device in the subsequent section. The program efficiently utilizes 419 KB of flash memory and 258 KB of RAM, which are well-suited for the resource constraints of the ESP32-S3 platform. After deploying the program on the ESP32-S3 board, we conducted real data tests to evaluate the duration of all steps. The specific times taken for each step are listed in Table 1 and Table 2. Table 1 provides the running time of the device utilized for the pre-processing of 3s raw data, which includes the FIR filter with an order number of 43, STE calculation for both 200 parameter windows and 400 parameter slice windows, and end-point detection algorithms. This value remained constant throughout the execution process. Meanwhile, table 2 illustrates the timing values for classifying the segmented audio extracted from the source data, including MFE calculation and inference via the machine learning model. The processing time for audio segments under 250 ms consistently remained at 97 ms, while longer segments, lasting between 750 and 1,000 ms, required approximately 356 ms across four consecutive recognition executions. These timing tests demonstrate that the integrated algorithms can be executed in real time on the ESP32-S3.

Table 1: Pre-processing time for 3s of raw data.

No.	Step	Time (ms)
1.	Band pass filter FIR	310
2.	STE calculating	45
3.	End-point detection	25
	Total:	380

Source: Authors, (2026).

Table 2: Inference for each window sound.

No.	Step	Time (ms)
1.	MFE calculating	84
2.	Model inference	13
	Total:	97

Source: Authors, (2026).

Furthermore, the prototype device has been successfully fabricated as shown in Figure 8. The device comprises an audio acquisition and processing module that is seamlessly connected to a display via UART. The display continuously shows the count of detected "Unhealthy" sounds, providing real-time feedback and facilitating prompt intervention. This integrated system highlights the practical viability of our solution for on-device poultry health monitoring under realworld conditions.



Figure 8: Prototype implementation device.
Source: Authors, (2026).

While previous studies [4-9] [16-21] have focused on recognizing and analyzing the health status of chickens through sound using local processing on computers, the results of this study demonstrate that the device can meet the requirements for detecting diseased chickens based on sound in real-time conditions and can be applied in practical scenarios with inexpensive processors with the support of the DSP algorithm and TinyML solution. Moving forward, we plan to conduct additional experiments and apply these datasets to real-world scenarios to further evaluate their detection accuracy. By integrating this research with advancements in IoT, we can create a comprehensive solution that empowers farmers to monitor and predict chicken diseases in real time, thereby enhancing animal health care and the quality of agricultural products.

V. CONCLUSIONS

In this study, an edge-computing solution is proposed to detect unhealthy chickens based on their vocalizations. This solution was developed using the DSP algorithm and TinyML, which can be implemented on a device with limited resources, such as a MCU. The endpoint detection method utilizes short-time energy, MFE extraction, and machine-learning model inference, and it was tested on a model system that uses augmented data on Edge Impulse for both the training and testing phases. The results of testing on ESP32-S3 showed that with 419 KB of flash memory and 258 KB of RAM, the device could detect unhealthy conditions in real time using TinyML, achieving 92.8% accuracy in the training phase and 90.58% accuracy in the testing phase for classifying healthy, unhealthy, and noisy sounds. The system also demonstrated a high level of accuracy in detecting diseased chickens, with an F1-score of 0.98. This research can be further explored to create more sophisticated automated poultry health-monitoring systems that can be applied in real-world conditions.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Minh Nguyen-Ngoc, Duan Luong-Cong.

Methodology: Author One and Author Two.

Implementation: Bien Nguyen-Quang, Minh Nguyen-Ngoc.

Discussion of results: Minh Nguyen-Ngoc, Bien Nguyen-Quang, Duan Luong-Cong.

Writing – Original Draft: Bien Nguyen-Quang.

Writing – Review and Editing: Minh Nguyen-Ngoc, Duan Luong-Cong.

Approval of the final text: Minh Nguyen-Ngoc, Bien Nguyen-Quang, Duan Luong-Cong.

VII. ACKNOWLEDGMENTS

This research is funded by Posts and Telecommunications Institute of Technology (PTIT).

VIII. REFERENCES

- [1] N. Li, Z. Ren, D. Li, and L. Zeng, "Review: Automated techniques for monitoring the behaviour and welfare of broilers and laying hens: towards the goal of precision livestock farming," *Animal*, vol. 14, no. 3, pp. 617–625, 2020, doi: 10.1017/S1751731119002155.
- [2] Rowe, Dawkins, and Gebhardt-Henrich, "A Systematic Review of Precision Livestock Farming in the Poultry Sector: Is Technology Focussed on Improving Bird Welfare?," *Animals*, vol. 9, no. 9, p. 614, Aug. 2019, doi: 10.3390/ani9090614.
- [3] C. L. Xu, Y. Z. Wang, J. Guo, J. X. Liu, and J. Feng, "Comparison of age-related differences in expression of antioxidant enzyme mRNA and activity in various tissues of pigs," *Comp Biochem Physiol B Biochem Mol Biol*, vol. 147, no. 3, pp. 445–451, Jul. 2007, doi: 10.1016/j.cbpb.2007.02.007.
- [4] K. Cuan, T. Zhang, Z. Li, J. Huang, Y. Ding, and C. Fang, "Automatic Newcastle disease detection using sound technology and deep learning method," *Computers and Electronics in Agriculture*, vol. 194, p. 106740, Mar. 2022, doi: 10.1016/j.compag.2022.106740.
- [5] S. Adebayo *et al.*, "Enhancing poultry health management through machine learning-based analysis of vocalization signals dataset," *Data in Brief*, vol. 50, p. 109528, Oct. 2023, doi: 10.1016/j.dib.2023.109528.

- [6] M. M. Ahmed, E. E. Hassanien, and A. E. Hassanien, "A smart IoT-based monitoring system in poultry farms using chicken behavioural analysis," *Internet of Things*, vol. 25, p. 101010, Apr. 2024, doi: 10.1016/j.iot.2023.101010.
- [7] T. S. Gunawan, M. F. Sabar, H. Nasir, M. Kartiwi, and S. M. A. Motakabber, "Development of Smart Chicken Poultry Farm using RTOS on Arduino," in *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, Kuala Lumpur, Malaysia: IEEE, Aug. 2019, pp. 1–5. doi: 10.1109/ICSIMA47653.2019.9057310.
- [8] J. Huang, W. Wang, and T. Zhang, "Method for detecting avian influenza disease of chickens based on sound analysis," *Biosystems Engineering*, vol. 180, pp. 16–24, Apr. 2019, doi: 10.1016/j.biosystemseng.2019.01.015.
- [9] L. Carpentier, E. Vranken, D. Berckmans, J. Paeshuyse, and T. Norton, "Development of sound-based poultry health monitoring tool for automated sneeze detection," *Computers and Electronics in Agriculture*, vol. 162, pp. 573–581, Jul. 2019, doi: 10.1016/j.compag.2019.05.013.
- [10] E.-S. M. Abdel-Kafy, S. E. Ibraheim, A. Finzi, S. F. Youssef, F. M. Behiry, and G. Provolto, "Sound Analysis to Predict the Growth of Turkeys," *Animals*, vol. 10, no. 5, p. 866, May 2020, doi: 10.3390/ani10050866.
- [11] M. Sadeghi, A. Banakar, M. Khazaei, and M. Soleimani, "An Intelligent Procedure for the Detection and Classification of Chickens Infected by Clostridium Perfringens Based on their Vocalization," *Rev. Bras. Cienc. Avic.*, vol. 17, no. 4, pp. 537–544, Dec. 2015, doi: 10.1590/1516-635X1704537-544.
- [12] A. Banakar, M. Sadeghi, and A. Shushtari, "An intelligent device for diagnosing avian diseases: Newcastle, infectious bronchitis, avian influenza," *Computers and Electronics in Agriculture*, vol. 127, pp. 744–753, Sep. 2016, doi: 10.1016/j.compag.2016.08.006.
- [13] R. S. Alonso, I. Sittón-Candanedo, Ó. García, J. Prieto, and S. Rodríguez-González, "An intelligent Edge-IoT platform for monitoring livestock and crops in a dairy farming scenario," *Ad Hoc Networks*, vol. 98, p. 102047, Mar. 2020, doi: 10.1016/j.adhoc.2019.102047.
- [14] B. McFee *et al.*, *librosa: Audio and Music Signal Analysis in Python*. 2015, p. 24. doi: 10.25080/Majora-7b98e3ed-003.
- [15] B. Bechtold, *soundfile: An audio library based on libsndfile, CFFI and NumPy*. Python. Accessed: May 06, 2024. [OS Independent]. Available: <https://github.com/bastibe/python-soundfile>
- [16] A. Mahdavian, S. Minaei, P. M. Marchetto, F. Almasganj, S. Rahimi, and C. Yang, "Acoustic features of vocalization signal in poultry health monitoring," *Applied Acoustics*, vol. 175, p. 107756, Apr. 2021, doi: 10.1016/j.apacoust.2020.107756.
- [17] K. Cuan, Z. Li, T. Zhang, and H. Qu, "Gender determination of domestic chicks based on vocalization signals," *Computers and Electronics in Agriculture*, vol. 199, p. 107172, Aug. 2022, doi: 10.1016/j.compag.2022.107172.
- [18] H. Purwins, B. Li, T. Virtanen, J. Schluter, S.-Y. Chang, and T. Sainath, "Deep Learning for Audio Signal Processing," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 2, pp. 206–219, May 2019, doi: 10.1109/JSTSP.2019.2908700.
- [19] A. Chowdhury and A. Ross, "Fusing MFCC and LPC Features Using 1D Triplet CNN for Speaker Recognition in Severely Degraded Audio Signals," *IEEE Trans. Inform. Forensic Secur.*, vol. 15, pp. 1616–1629, 2020, doi: 10.1109/TIFS.2019.2941773.
- [20] S. Hymel *et al.*, "Edge Impulse: An MLOps Platform for Tiny Machine Learning," Apr. 28, 2023, *arXiv*: arXiv:2212.03332. Accessed: May 27, 2024. [Online]. Available: <http://arxiv.org/abs/2212.03332>
- [21] K. Cuan, T. Zhang, J. Huang, C. Fang, and Y. Guan, "Detection of avian influenza-infected chickens based on a chicken sound convolutional neural network," *Computers and Electronics in Agriculture*, vol. 178, p. 105688, Nov. 2020, doi: 10.1016/j.compag.2020.105688.