# Selective Audio Recording Device for Wildlife Research Using Embedded Machine Learning

Julia Azziz*, Josefina Lema*, Leonardo Steinfeld*, Emiliano Acevedo* and Martín Rocamora*†
*Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
†Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
Email: {jazziz, jlema, leo, eacevedo}@fing.edu.uy, martin.rocamora@upf.edu

***Abstract*—Wildlife monitoring through sound recording has become an essential tool in ecological research. However, challenges such as limited power and memory constraints hinder large-scale, long-term deployment of monitoring devices. To address these limitations, this paper presents a novel wildlife monitoring device that integrates embedded machine learning (ML) for event-triggered recording. This system captures only relevant sounds, leading to a more efficient memory usage and power consumption than the traditional fixed-schedule scheme, and a significantly larger percentage of useful data collected. The device features a low-cost, low-power hardware design equipped with a digital microphone, dual MicroSD storage and a flexible power system. Its embedded ML component enables real-time audio classification and selective recording triggered by specific acoustic events. Preliminary testing using a prototype device demonstrated effective detection of penguin vocalizations, achieving an average current intake ranging from 4.06 to 6.02 mA, depending on the operational mode. This enables the device to be powered by a small, cost-effective rechargeable battery and solar power, supporting near-perpetual operation. The proposed system represents a step forward in deploying low-cost, low-power scalable devices for acoustic wildlife monitoring.**

***Index Terms*—wildlife monitoring, sound recording, embedded machine learning, low-power, real-time audio classification**

## I. Introduction

The monitoring of wildlife is essential for understanding biodiversity, ecosystem dynamics, and the effects of environmental changes. Among the various monitoring techniques, sound recording has proven to be a powerful tool, allowing researchers to capture a wide range of ecological interactions, from animal calls to environmental sounds that are indicative of habitat conditions. However, deploying sound recording devices in the wild presents several challenges, including limited power resources, high operational costs, and the sheer volume of unfiltered data generated. These challenges make it difficult for long-term and large-scale studies to capture meaningful data without extensive manual intervention.

Traditional monitoring approaches use passive acoustic monitoring (PAM) devices that continuously capture audio data on a fixed schedule [1]. For this task, several commercial devices are available. Among the resources available to guide the selection of PAM equipment, the World Wildlife Fund provides a comprehensive list of hardware and software components in its guidelines for PAM in ecology and conservation [2]. According to this guide, the most commonly used commercial systems are the Song Meter by Wildlife Acoustics [3] and the BAR series by Frontier Labs [4], both highly regarded for their recording quality. These commercial PAM devices support high-fidelity recording, which aligns well with the needs of bioacoustic monitoring projects. However, their high cost, often exceeding 800 USD per unit, can be prohibitive for large-scale deployments where many recording points are required. In response to this, the AudioMoth [5], developed by Open Acoustic Devices, offers a more affordable alternative at only 97 USD. With comparable features, including open-source support, it provides flexibility for adaptation and integration into custom monitoring projects.

In addition, the AudioMoth offers a triggered recording mode, where audio samples are only recorded when triggered by a specific event. This trigger can be either the signal amplitude or the response within a specific frequency band surpassing a given threshold. However, such triggers are inherently limited, as they are not necessarily indicative of events of interest. For example, loud ambient sounds may exceed the amplitude threshold despite not being relevant to the study. As a result, these trigger mechanisms may still generate substantial amounts of unnecessary data.

In this paper, we introduce a novel design for a wildlife sound recording device that utilizes embedded machine learning (ML) to automatically detect and record only relevant sounds. By integrating an ML model into the device, we can minimize unnecessary recordings, thereby extending the device's operational lifespan and optimizing memory usage. This selective recording method ensures that only essential data is captured, reducing the need for post-processing while conserving battery life. We present the design and implementation and analyze preliminary measurements that showcase the device's performance.

This work is organized as follows: Section II presents the overall system architecture, describing all hardware, firmware and machine learning components. Section III presents experimental results, including model performance metrics and power consumption measurements. Section IV further analyses the current implementation and delves into future work. Finally, Section V concludes the paper.

## II. System description

The device was designed to balance low-power operation, efficient memory usage and reliable audio processing. Given

the constraints of continuous field deployment, key design considerations included optimizing power consumption, ensuring enough memory for high-quality audio data and meeting the timing requirements for real-time activity detection. The following sections outline the hardware, firmware, and machine learning model used in the device.

### A. Hardware

Figure 1 displays the proposed overall hardware architecture. The device is based on an EFR32MG24 microcontroller, chosen primarily due to its integrated Bluetooth Low Energy (BLE) capabilities and hardware accelerator. The system employs a digital ICS-43434 microphone and stores audio data using direct memory access (DMA), in order to offload the transfer process from the microcontroller. Among the microphone's specifications, we highlight its low power mode, which it can enter automatically when the sampling frequency is between 6.25 and 18.75 kHz.

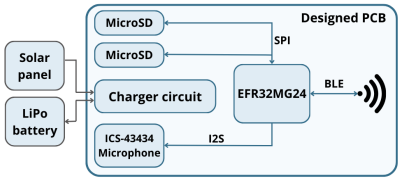


Fig. 1. Diagram showing the overall hardware architecture.

The board design includes two MicroSD card slots with separate SPI and power supply lines. This dual-slot configuration supports the use of either one or both cards; when the first one reaches capacity, data storage automatically switches to the second one. This setup helps prevent memory shortages in long-term deployments, ensuring continuous data recording.

A charging circuit was integrated into the PCB, designed around Texas Instruments' BQ24074 battery charger and specifically designed for flexibility in power sources. The system is primarily designed for use with a 1 W solar panel, charging a 3.7 V, 2500 mAh battery for off-grid functionality. However, the circuit also includes a USB input option, allowing for operation when solar power is unavailable. The USB input is only used as a power source, as the board is programmed via the SWD interface with a J-Link debugger.

To further minimize power usage, the device includes hardware-level power management for components like the SD card interface, allowing selective power control to reduce energy consumption when data storage is not active. A reed switch is used to activate BLE functionality, triggering a digital enable signal when a magnet is brought near.

Figure 2 shows the designed 2-layer PCB, which has a size of 58.4 x 51.2 mm. Components were only placed on the top side of the board. The building cost for this device is 53 USD.

### B. Firmware

Firmware was developed using Silicon Labs' software development kit (SDK) and the TensorFlow Lite for Microcontrollers (TFLM) C++ library. The core functionality is the

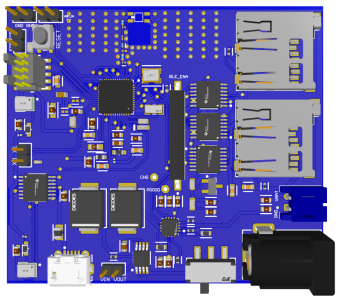

Fig. 2. Rendered 3D view of the designed PCB.

main audio processing loop, which can be divided into three stages: feature extraction, model inference and file recording.

Audio data is continuously acquired from the microphone and stored in a ring buffer in chunks of 512 bytes using DMA. This allows for parallel execution of the main audio processing task, which begins once a full buffer transfer is completed and the microphone starts to capture the next one. Immediately after completed, the last audio buffer is copied so that DMA can continue filling the original ring buffer, and its features are extracted using the TFLM library (as will be detailed in Section II-C). The buffer size is left as a configurable parameter, and its length will determine the maximum latency for a full iteration of the processing loop:

$$T_{buffer} > T_{features} + T_{inference} + T_{writing}. \quad (1)$$

The file recording sequence was implemented using a double hysteresis mechanism, to ensure that relevant audio is captured with sufficient context while avoiding the capture of spurious events. The recording process starts when $n$ 1-second frames activity are detected, which helps filter out transient sounds or noise so that only sustained activity triggers the recording. Similarly, it waits for $m$ consecutive frames of no activity to stop recording. Figure 3 shows a timing diagram of the recording sequence, using $n = 1$ s and $m = 2$ s. The recorded file includes not only the full activity period, which covers the initial $n$ frames, but also incorporates the last frame of preceding inactivity and the first frame of subsequent inactivity.

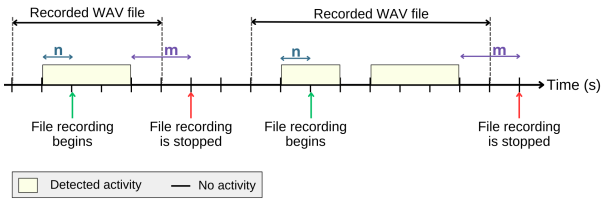


Fig. 3. Diagram showing the timing of the recording sequence, using $n = 1$ s and $m = 2$ s.

To ensure accurate timekeeping and maintain continuity after a reset, the device relies on a low-power RTC module integrated into the EFR32MG24 that periodically records the current epoch to a non-volatile flash partition. Upon reboot,

the system loads this timestamp and reprograms the RTC, ensuring that all recorded files have the correct timestamp. This introduces a trade-off between time precision after a reset and the frequency at which current time is stored.

### C. Machine learning model

The EFR32MG24's hardware accelerator supports acceleration for all TensorFlow operators used in a convolutional neural network (CNN), one of the most commonly used architectures in audio classification. To take advantage of this computational offload, the model should be either developed using TensorFlow or converted to the TFLite format after training. The resulting binary file is then converted to a C++ array, which is loaded by the TFLM API in the system initialization function. The firmware was designed with a model-agnostic architecture, isolating the machine learning model from the core functionality. This enables the user to replace the model file based on the specific target, sound, or species of interest without having to modify application code.

Inference time and RAM usage represent the main constraints when developing the model. As shown by Eq. 1, and considering that inference is the most computationally expensive stage of the loop, the length of the audio buffer can be taken as an upper limit for inference time. Inference must be completed within this period to ensure real-time responsiveness and no data loss. Additionally, RAM limitations require a compromise in model complexity, especially for higher sampling frequencies that demand larger buffer sizes.

The system was designed to use log-mel spectrograms as input features. Given that the specific parameters will vary according to the target species, the configuration will be loaded from the TFLite model's metadata, which can be specified by the user and is embedded into the binary file during conversion. When training the model, features should be computed using the Python wrapper for the TFLM log-mel spectrogram function, provided by the MLTK. This ensures that training and validation data are consistent with the 8-bit feature arrays generated by the device during deployment, avoiding significant loss in performance after the trained model is further quantized.

## III. Experiments and results

The feasibility of the proposed concept was evaluated in terms of both model performance and power consumption. For preliminary testing, we used Silicon Labs' xG24 Development Kit (DK) in place of the custom-designed PCBs, which were still under production. The DK features the same microcontroller and microphone as the target hardware, allowing us to evaluate basic functionality.

### A. Experimental setup

To demonstrate the effectiveness of the device for wildlife sound detection, penguin vocalizations were chosen as the target audio class. For the binary audio classification task a ResNet18 CNN was selected, due to its suitability for audio classification and reduced size [6]. The input log-mel spectrograms were configured with a sampling rate of 16 kHz, a window length of 30 ms and an overlap of 10 ms, yielding 96 mel bands. The device was set to record with an activation threshold of $n = 1$ s and and inactivity threshold of $m = 2$ s. A 128 GB MicroSD card was connected to the DK through a breakout board.

### B. Model performance

To evaluate model performance across different quantization levels, accuracy, memory usage and inference time were selected as the key metrics.

Three different quantization methods were compared. The first method is full integer quantization, which converts all parameters and operations to int8 format. Similarly, dynamic range quantization converts weights and activations to 8-bit integers while maintaining floating-point outputs. Finally, 16x8 quantization results in 16-bit activations with 8-bit weights. Despite models converted using 16x8 quantization not being supported by the hardware accelerator, this method was evaluated in order to assess the impact of quantizing activations. Table I presents a comparative overview of the four models, in order to evaluate key performance metrics: accuracy, memory usage and inference time. Memory usage has two components: model size, which represents static parameter storage in flash memory, and runtime memory, which is the necessary RAM allocation required. Memory and inference time measurements were estimated using Silicon Labs' Machine Learning Tool Kit (MLTK) [7], which allows for direct on-device profiling.

TABLE I
PERFORMANCE METRICS ACROSS DIFFERENT MODEL QUANTIZATION METHODS, ESTIMATED WITH THE MLTK.

| Quantization | Size | Runtime memory (kB) | Inference time (ms) | Accuracy |
|---|---|---|---|---|
| None | 1.3 MB | 196.7 | 763.2 | 0.948 |
| Full integer | 370.9 kB | 77.3 | 114.7 | 0.947 |
| Dynamic range | 359.6 kB | 196.7 | 763.2 | 0.946 |
| 16x8 | 383.5 kB | 107.3 | 767.6 | 0.947 |

Full integer quantization shows the most substantial reduction in runtime memory usage, cutting it by nearly 61% compared to the original model. This method also achieves the fastest inference time, reducing latency by approximately 85%. Dynamic range quantization achieves the smallest memory footprint, but it fails to improve inference speed due to its reliance on floating-point operations. Meanwhile, 16x8 quantization yields moderate reductions in memory usage but offers no improvement in inference time due to its incompatibility with hardware acceleration. Table I also shows that the accuracy of all quantization methods remains largely consistent, with only marginal variations observed.

Considering the strict RAM limitations and the timing requirements described in previous sections, the fully quantized model is the most efficient overall choice.

### C. Power consumption

Nordic's Power Profiler Kit II was used to measure current consumption, using the fully quantized model. Given that these measurements were taken during preliminary testing on the DK, they do not fully represent the power characteristics of the final custom hardware. However, these measurements are still useful as they demonstrate how specific hardware or firmware modifications impact overall power usage. For instance, reducing current draw by deactivating the MicroSD card is expected to yield similar savings on the final device, as the effect of such actions on current consumption remains consistent across configurations.

The average current intake during normal operating conditions was measured to be 6.02 mA when writing a file and 4.06 mA when not writing. This allows the device to be powered by a small, cost-effective rechargeable battery and solar power, supporting near-perpetual operation. Table II provides a breakdown of the duration and average current consumption for all states in one full iteration of the main recording loop. This includes the baseline current intake from DK-specific hardware. Table II also demonstrates that the complete iteration takes less than one second, since the recording and DMA are executed in parallel, leaving headroom to employ a more complex model with a longer inference time while still maintaining correct operation.

TABLE II
CURRENT INTAKE MEASUREMENTS FOR DIFFERENT STATES INVOLVED IN THE MAIN RECORDING LOOP.

| Description | Duration (s) | Average current (mA) |
|---|---|---|
| Recording + DMA | Continuous | 3.55 |
| Feature extraction[1] | $T_{features} = 0.246$ | 3.86 |
| Running inference[1] | $T_{inference} = 0.279$ | 4.66 |
| Writing to SD card[1] | $T_{writing} = 0.109$ | 23.2 |

[1] Measurements also include the *recording + DMA* current.

## IV. FURTHER ANALYSIS AND FUTURE WORK

Future work will focus on conducting extensive field testing to properly validate the device's performance, using the designed PCB instead of the DK prototype. We expect this will yield more accurate measurements, particularly in terms of power consumption. Due to the DK and the custom board having fundamental differences in terms of hardware components, with the DK containing numerous additional features and peripherals, power consumption is expected to decrease significantly. To complement laboratory testing, we plan on deploying the device in proximity to a penguin colony for three weeks. By installing the device near active penguin habitats, we aim to assess both the device's ability to selectively record vocalizations and its battery lifespan in real operating conditions. A key focus will be evaluating the precision-recall trade-off, which determines how well the device captures what it is supposed to record, and how accurately it filters out irrelevant sounds. With this goal, we also plan to compare the performance of the adaptive recording system with that of a fixed-schedule recorder, which will be installed alongside the proposed device.

In addition, future iterations of the device are intended include adaptive power management techniques, such as more refined duty cycling and advanced sleep modes, ensuring energy efficiency while maintaining a reliable detection and recording schedule. Furthermore, Bluetooth Low Energy (BLE) connectivity will be implemented, enabling users to consult status variables in real time to monitor device behavior and gather usage statistics. This will allow for more comprehensive debugging and performance analysis during field deployments, as well as collecting statistical data about the target species' behavior.

## V. CONCLUSIONS

This paper presents a novel wildlife monitoring device that integrates embedded machine learning to enable event-triggered recording of relevant audio, significantly reducing power consumption and memory usage. The design's modular architecture allows users to easily adapt it for different target species by replacing the embedded model without modifying the firmware. Preliminary results demonstrate the feasibility of the proposed approach, showcasing effective real-time audio classification for penguin vocalizations. Measurements taken using a prototype built around the xG24 DK yield a maximum average current intake of 6.02 mA, which is taken as an upper limit for the designed board's actual power consumption.

Future work will focus on extensive field testing to evaluate the device's performance under real conditions. Additional optimizations will include further reducing power consumption, integrating alternative energy sources such as solar panels and implementing BLE connectivity. The proposed system represents a promising step toward scalable, low-cost, efficient solutions for wildlife acoustic monitoring.

## REFERENCES


[1] L. Sugai, C. Desjonquères, T. Silva, and D. Llusia, "A roadmap for survey designs in terrestrial acoustic monitoring," *Remote Sensing in Ecology and Conservation*, vol. 6, 11 2019.

[2] E. Browning, R. Gibb, P. Glover-Kapfer, and K. E. Jones, "Passive acoustic monitoring in ecology and conservation," WWF-UK, Tech. Rep. 1(2), 2017. [Online]. Available: https://www.wwf.org.uk/sites/default/files/2019-04/Acousticmonitoring-WWF-guidelines.pdf

[3] Wildlife Acoustics, Inc., *Song Meter User Guide*, June 2024. [Online]. Available: https://www.wildlifeacoustics.com/uploads/user-guides/SM4-USER-GUIDE-EN-2024-06-11.pdf

[4] Frontier Labs, *Bioacoustic Audio Recorder - Long Term User Guide*, 2017.

[5] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "AudioMoth: A low-cost acoustic device for monitoring biodiversity and the environment," *HardwareX*, vol. 6, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468067219300306

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[7] S. Labs, "Silicon Labs Machine Learning Toolkit (MLTK)." [Online]. Available: https://github.com/SiliconLabs/mltk