

# CONESCAPANHONDURAS2025paper6.pdf

 Institute of Electrical and Electronics Engineers (IEEE)

---

## Document Details

### Submission ID

trn:oid:::14348:477787975

### Submission Date

Jul 31, 2025, 11:57 PM CST

### Download Date

Aug 1, 2025, 1:06 PM CST

### File Name

CONESCAPANHONDURAS2025paper6.pdf

### File Size

1.3 MB

5 Pages





2,091 Words

12,677 Characters




# 3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Match Groups

-  **3 Not Cited or Quoted 2%**  
Matches with neither in-text citation nor quotation marks
-  **1 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **2 Missing Citation 1%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

## Top Sources

- 3%  Internet sources
- 3%  Publications
- 0%  Submitted works (Student Papers)

## Integrity Flags





### 0 Integrity Flags for Review

No suspicious text manipulations found.




Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

## Match Groups

-  **3 Not Cited or Quoted 2%**  
Matches with neither in-text citation nor quotation marks
-  **1 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **2 Missing Citation 1%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

## Top Sources

- 3%  Internet sources
- 3%  Publications
- 0%  Submitted works (Student Papers)

## Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

- 1** **Internet**  
**arxiv.org** 1%
- 2** **Publication**  
**Yanchi Dong, Tianyu Jia, Kaixuan Du, Yiqi Jing et al. "A Model-Specific End-to-End ...** <1%
- 3** **Internet**  
**www.scribd.com** <1%
- 4** **Internet**  
**export.arxiv.org** <1%
- 5** **Internet**  
**www.isca-archive.org** <1%

# Voice-Activated IoT Devices: Deploying Real-Time Speech Recognition on Arduino via Edge Impulse

**Abstract**—This paper addresses the implementation of speech recognition on embedded hardware by combining digital signal processing techniques, machine learning models, and edge computing. The proposed system will leverage Mel-Frequency Cepstral Coefficients (MFCC) for feature extraction and quantization techniques to reduce the computational burden while maintaining accuracy. The model will be trained using Edge Impulse and will be deployed on an Arduino-based platform, evaluating real-time performance under different environmental conditions. The impact of digital signal preprocessing will be evaluated in terms of accuracy, inference latency, and power consumption.

**Index Terms**—Digital signal processing, machine learning, edge computing.

## I. INTRODUCTION

Speech recognition has become a fundamental tool in modern IoT applications, enabling natural human-device communications in industrial automation, smart homes and healthcare wearables. However, deploying real-time speech recognition on low-power consumption microcontrollers presents significant challenges due to computational constraints, limited memory, and power efficiency requirements. Most existing solutions rely on cloud computing which introduces latency, privacy concerns and network connectivity dependence. To address these limitations this paper explores the feasibility of implementing real-time speech recognition directly on an Arduino Nano 33 BLE Sense board using digital signal processing techniques and optimized machine learning models.

## II. STATE OF THE ART

Recent advancements in speech recognition and edge computing have focused on optimizing accuracy, latency, and energy efficiency for resource-constrained devices. This section reviews key developments in three areas: edge-based speech recognition, TinyML optimizations, and digital signal processing in microcontroller integration.

### A. Edge-based speech recognition

Early work in embedded speech recognition prioritized keyword spotting with lightweight models. Warden introduced the Google Speech Commands Dataset and a CNN-based KWS model achieving 90 accuracy on 12 commands, but required 250 KB RAM, limiting deployment to mid-tier microcontrollers [1]. Subsequent efforts optimized architectures includes MCUNet co-designed neural networks and memory allocation for microcontrollers, enabling 80 percent accuracy on 10 commands with 40 KB RAM [2]

and RNN-based models reduced latency to 120 ms using temporal convolutions but suffered from high compute costs (1.2 MFLOPS) [3].

While these works demonstrated feasibility, they relied on raw audio waveforms, ignoring digital signal processing for feature extraction that could reduce model complexity.

### B. Tiny ML Optimization

Model compression techniques have enabled ML on microcontroller 8-bit post-training quantization reduced ResNet-10 speech models to 50 KB with less than 3 percent accuracy loss [4].

### C. Digital signal processing

These works focused on isolated DSP or ML layers, failing to holistically optimize the audio pipeline from raw data to actuation like MFCC Acceleration using CMSIS-DSP libraries optimized FFT and filter operations on Cortex-M cores, cutting MFCC latency by 50 percent vs. vanilla C [5]. STM32Cube.AI integrated CMSIS-DSP with TensorFlow Lite, achieving 80 ms end-to-end latency for KWS [6].

## III. PROPOSED SYSTEM OVERVIEW

This section outlines the essential components that might be used in the project, including microcontrollers, development tools, and software frameworks.

### A. Hardware

- Arduino Nano 33 BLE Sense: equipped with an Arm Cortex-M4 processor and an integrated microphone, ideal for low-power machine learning applications.
- External supply source: 3.3 V Battery
- Relay Module: 3.3 V relay, according on the Arduino output voltage. This module is demonstrating the real-world application of the speech recognition system. Speech commands can activate or deactivate the relay.

### B. Software and development tools

- Edge Impulse: A platform for training and optimizing machine learning models for embedded devices. It provides tools for data acquisition, feature extraction, and deployment.
- Arduino IDE: The primary development environment for programming the microcontroller and integrating the trained model.

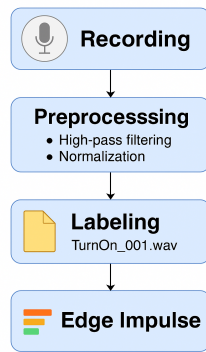


Fig. 1. Data acquisition and Preprocessing workflow

- TensorFlow Lite for Microcontrollers: A lightweight version of TensorFlow optimized for running deep learning models on microcontrollers.
- MATLAB: Very useful for data acquisition and filtering.

#### IV. DATA ACQUISITION AND PREPROCESSING

The acquisition of voice commands was conducted using MATLAB and the built-in microphone of a standard laptop. A custom MATLAB script was developed to streamline the collection of labeled audio samples. The script allows recording multiple instances of a specific command while automatically applying digital signal processing techniques for enhancement.

Each recording session involves:

- Sampling audio at 16 kHz (Same frequency as the Arduino's integrated microphone) with 16-bit resolution.
- Applying a 4th-order high-pass Butterworth filter with a cutoff frequency of 300 Hz to remove low-frequency noise and improve speech clarity.
- Performing normalization to ensure uniform signal amplitude across all samples.
- Saving the audio files in .wav format using a sequential naming convention.

These steps not only automate the data acquisition pipeline but also ensure consistent and high-quality input for training the neural network. The processed dataset was then uploaded to Edge Impulse Studio, where each sample was labeled accordingly for supervised learning. The overall flow of data acquisition and preprocessing is shown in Fig. 1.

#### V. MACHINE LEARNING MODEL DEVELOPMENT AND TRAINING

##### A. Dataset construction

A total of six voice commands were selected for training the speech recognition system. These commands are representative of typical control phrases used in voice-activated IoT environments, such as "Encender", "Apagar",

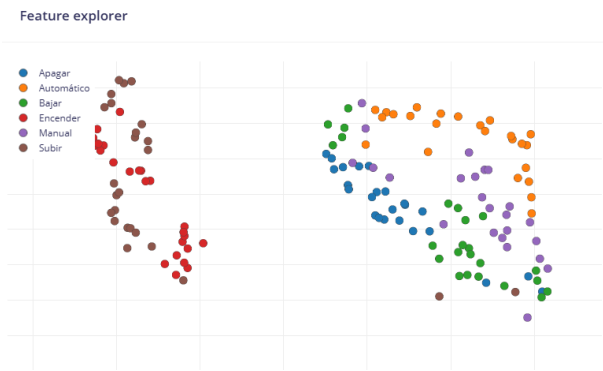


Fig. 2. Feature Extraction in Edge Impulse

"Subir", "Bajar", "Automático", and "Manual".

Each command was recorded 30 times, resulting in a dataset of 180 labeled audio samples. All recordings were captured at a sampling rate of 16 kHz and 2 seconds length using the custom MATLAB script that was mentioned in the previous section.

Once preprocessed, the dataset was uploaded to Edge Impulse Studio, where it was automatically split per class into 80% for training and 20% for validation. The splitting process was randomized to ensure that each class was evenly represented in both subsets, reducing bias and improving model generalization.

##### B. Feature Extraction

To effectively transform raw audio signals into a format suitable for machine learning, DSP techniques were applied during the preprocessing in Edge Impulse. The primary objective of this stage was to extract compact and informative features that represent the temporal and spectral characteristics of human speech for the recorded commands.

The resulting MFCC features served as the input to the neural network model. This approach ensured that the system remained computationally efficient while maintaining high recognition accuracy, which is critical for real-time inference on memory-constrained microcontrollers. We can see a summary of all extracted features for each command in Fig. 2.

##### C. Model Architecture

Once the MFCC features were extracted, a convolutional neural network (CNN) architecture was selected due to its effectiveness in recognizing spatial patterns in time-frequency representations. The architecture was selected using the default parameters and layers provided by Edge Impulse platform as is shown in Fig. 3.

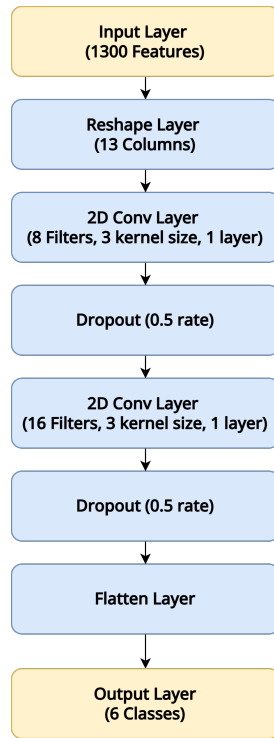


Fig. 3. Neural Network Model

#### D. Model Training and Optimization

The proposed CNN architecture was trained on the Edge Impulse platform using the preprocessed MFCC features with the following hyperparameters:

- Number of training cycles: 100
- Learning rate: 0.005

To enhance generalization and improve robustness to noise and signal variation, data augmentation techniques were enabled during training. These included:

- Random noise: Low
- Time masking: Low
- Frequency masking: Low

The trained model achieved an overall validation accuracy of 89.7%, with a cross-entropy loss of 1.39. Performance metrics on the validation set further include:

TABLE I  
METRICS' PERFORMANCE ON VALIDATION SET

Metric	Score
Accuracy	89.7%
Weighted average precision	0.91
Weighted average recall	0.90
Weighted average F1-score	0.90
Area under ROC curve (AUC)	0.97

	APAGAR	AUTOMÁTICO	BAJAR	ENCENDER	MANUAL	SUBIR
APAGAR	100%	0%	0%	0%	0%	0%
AUTOMÁTICO	0%	100%	0%	0%	0%	0%
BAJAR	0%	0%	80%	0%	20%	0%
ENCENDER	0%	0%	0%	100%	0%	0%
MANUAL	14.3%	0%	0%	0%	85.7%	0%
SUBIR	0%	0%	0%	25%	0%	75%
F1 SCORE	0.80	1.00	0.89	0.91	0.86	0.86

Fig. 4. Confusion matrix for the classifier

<b>EON™ Compiler</b> Same accuracy, 23% less RAM, 33% less ROM.				
Quantized (int8)		MFCC	CLASSIFIER	TOTAL
LATENCY	475 ms.		66 ms.	541 ms.
RAM	21.1K		15.3K	21.1K
FLASH	-		40.8K	-
ACCURACY				-
Unoptimized (float32)		MFCC	CLASSIFIER	TOTAL
LATENCY	475 ms.		1,080 ms.	1,555 ms.
RAM	21.1K		53.3K	53.3K
FLASH	-		65.0K	-
ACCURACY				89.71%

Fig. 5. Performance of quantized and non-quantized model

The confusion matrix from Fig. 4. highlights strong classification performance in most classes, with 100% accuracy in "Apagar", "Automático", and "Encender", while classes like "Bajar", "Manual", and "Subir" exhibit minor confusion-likely due to similarities in pronunciation or limited training data. These insights suggest potential improvements through further data augmentation or model fine-tuning.

#### VI. DEPLOYMENT ON MICROCONTROLLER

Once training is complete, the next step is to quantize the model. For this purpose, we use the deployment options available in Edge Impulse along with the EON Compiler, which optimizes RAM and ROM usage. The quantized model is exported as an Arduino library that includes all the necessary code for inference and feature extraction, allowing it to run on the board.

Quantization significantly reduces RAM and Flash memory usage and shortens inference time compared to the unoptimized model. A comparison between both models is shown in Fig. 5.

The Arduino board runs the inference example provided in the library generated by Edge Impulse. This code executes the trained model and displays both the inference results and the time required for DSP and classification directly on the board via the serial monitor, as shown in Figure 6.

#### VII. JUSTIFICATION OF THE PROJECT

Traditional voice recognition systems, such as Amazon Alexa, Google Assistant, or Apple Siri, rely heavily on cloud-based infrastructure to perform speech processing and command interpretation. While these systems offer high accuracy and natural language understanding, they also

```
Recording done
Predictions (DSP: 273 ms., Classification: 91 ms., Anomaly: 0 ms.):
Apagar: 0.00781
Automático: 0.02344
Bajar: 0.95703
Desconocido: 0.00000
Encender: 0.00000
Manual: 0.00781
Subir: 0.00000
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 273 ms., Classification: 91 ms., Anomaly: 0 ms.):
Apagar: 0.00391
Automático: 0.11328
Bajar: 0.82422
Desconocido: 0.00000
Encender: 0.00000
Manual: 0.00000
Subir: 0.05859
Starting inferencing in 2 seconds...
```

Fig. 6. Arduino's serial monitor with CNN classification results

present several limitations for integration into low-power embedded or industrial environments:

- Internet dependency: Most commercial systems require a constant network connection to function, making them unsuitable for offline or remote use cases.
- Privacy concerns: Audio data must be transmitted to external servers for processing, raising concerns in sensitive applications where user data confidentiality is critical.
- Hardware and infrastructure cost: These solutions often depend on specialized hardware or proprietary ecosystems, which increases cost and limits flexibility for custom implementations.

In contrast, the proposed system offers a fully embedded and lightweight alternative to voice command recognition, specifically designed to run locally on resource-constrained microcontrollers such as the Arduino Nano 33 BLE. By leveraging digital signal processing and TinyML techniques, the system:

- Executes inference directly on the device without internet connection.
- Offers lower latency due to local processing.
- Ensures greater data privacy by avoiding cloud transmission.
- Can be implemented at a fraction of the cost using open-source tools and low-power hardware.

## VIII. SCOPE AND LIMITATIONS

### A. Scope

This study focuses on deploying a real-time Spanish speech recognition system for IoT applications using resource-constrained hardware. Key aspects of the project include:

- Implementation of a lightweight CNN model optimized via 8-bit quantization, targeting the Arduino Nano 33 BLE Sense.
- Use of MFCC-based feature extraction and digital signal preprocessing (high-pass filtering and normalization) to enhance speech clarity and reduce computational overhead.

- Evaluation of six voice commands ("Encender," "Apagar," "Subir," "Bajar," "Automatico," "Manual") representative of IoT control scenarios.
- Testing under controlled environmental conditions to measure accuracy, latency, and power consumption.

### B. Limitations

The proposed system has several constraints that affect its generalizability and performance:

- Hardware Constraints: The limited RAM (256 KB) and flash memory (1 MB) of Arduino restrict the complexity of the model and the size of the dataset.
- Dataset Size: Only 180 samples (30 per command; 24 for training and 6 for validation) were used, which may lead to overfitting and reduced robustness to voice variations (e.g., accents, pitch).
- Language and Vocabulary: The model supports only six Spanish commands, limiting its applicability to other languages or broader vocabularies without retraining. Also, the commands must be phonetically distinguishable and short expressions.

## IX. CONCLUSION AND FUTURE WORK

This paper presents the initial development of a voice-controlled IoT system that integrates digital signal processing techniques and a lightweight convolutional neural network for speech recognition on an embedded device. Through the use of MATLAB for data acquisition, MFCC-based feature extraction, and Edge Impulse for model training and deployment, a functional end-to-end pipeline has been established.

The proposed system currently supports six distinct voice commands and demonstrates reliable performance, achieving a validation accuracy of 89.7%.

However, this work is still in progress and further refinement is needed. Key areas for improvement include increasing the size and diversity of the dataset, enhancing the robustness of the model in noisy environments, and optimizing latency and memory usage on the microcontroller. Future iterations will also explore the inclusion of additional commands and real-time testing in more realistic conditions.

This preliminary version provides a strong foundation for the final system and sets a clear path for the remaining phases of development.

## REFERENCES

- [1] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," arXiv:1804.03209, 2018.
- [2] J. Lin et al., "MCUNet: Tiny Deep Learning on IoT Devices," NeurIPS, 2020.
- [3] Y. Zhang et al., "Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," INTERSPEECH, 2021.

4

- [4] P. Warden, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," MLSys, 2020.
- [5] ARM, "CMSIS-DSP: Optimized DSP Libraries for Cortex-M," ARM Developer, 2023.
- [6] STMicroelectronics, "STM32Cube.AI: AI at the Edge," STM32 Technical Documentation, 2023.