International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Secure Offline Voice Authentication and Command System Using TinyML for Edge Devices

Gangu V B, Preetham A G, Sayyad Karjagi, A J Tejas, Mrs. Harshitha K U
Department of Electronics and Communication Engineering
Kalpataru Institute of Technology, Tiptur, Karnataka, India

***Abstract*—Voice-based interfaces provide a convenient method of interacting with embedded systems. However, conventional cloud-connected voice assistants depend on network availability and may transmit sensitive voice data to remote servers. This paper presents the design of a secure offline voice authentication and command system using Tiny Machine Learning (TinyML) for edge devices. The proposed prototype uses an ESP32-family microcontroller and an INMP441 I^2S digital microphone to acquire voice input. The captured audio is processed locally, and voice features are extracted using Mel-Frequency Cepstral Coefficients (MFCC). A machine-learning model developed using the Edge Impulse workflow is intended to distinguish an authorized voice from an unauthorized voice. Based on the model output and the configured decision logic, the controller permits or denies the requested operation. For the access-control demonstration, a relay module interfaces with a 12 V solenoid door lock, while LEDs provide status indication. The principal objective is to perform authentication and command processing on the edge device without requiring an Internet connection during operation. This paper describes the system architecture, hardware and software design, audio processing workflow, TinyML model-development methodology, and functional evaluation procedure. Quantitative accuracy, false-acceptance rate, and false-rejection rate must be reported using the final independent test results obtained from the implemented model.**

***Index Terms*—TinyML, voice authentication, speaker recognition, offline authentication, ESP32, INMP441, MFCC, Edge Impulse, embedded systems, access control.**

I. Introduction

A. Background

Voice is a natural and convenient means of human interaction. Voice-controlled embedded devices can be used in access-control systems, home automation, assistive devices, and industrial applications. A voice-based access-control system can reduce the need for physical keys or passwords by allowing a user to interact with a device through speech.

Many voice-enabled applications use cloud-based speech-processing services. Although these services can provide powerful recognition capabilities, their operation generally depends on network connectivity and may require voice recordings to be transmitted to external servers. This creates challenges in areas with limited connectivity and in applications where local processing and privacy are important.

Tiny Machine Learning (TinyML) enables machine-learning inference on resource-constrained microcontrollers. By deploying a compact model directly on an embedded device, voice-related processing can be performed locally. The proposed system applies this approach to offline voice authentication and command-controlled access.

B. Problem Statement

Conventional voice-controlled access systems may depend on an Internet connection or external processing services. Such dependence can affect availability when network connectivity is poor or unavailable. In addition, a basic voice-command detector may respond to a spoken command without verifying whether the speaker is authorized.

The problem addressed in this work is the development of an embedded prototype that captures voice input, performs local voice-based authorization using a TinyML model, and controls an access mechanism according to the authentication decision. The system is intended to operate without Internet connectivity during authentication. Its reliability depends on the quality and diversity of the voice dataset, model performance, microphone conditions, and the decision logic implemented in the controller.

C. Motivation

The motivation for this work is to investigate a low-cost embedded approach to voice authentication that does not require cloud access during normal operation. Local inference can reduce dependence on network availability and keep the voice-processing workflow within the device.

The use of a digital microphone, a microcontroller, and a compact machine-learning model provides a practical platform for studying the integration of audio acquisition, feature extraction, inference, and hardware control. The project also provides an opportunity to evaluate the limitations of voice-based authentication in a real embedded environment.

D. Objectives

The main objectives of the proposed system are:

- To acquire voice input using an INMP441 I^2S digital microphone.
- To configure the ESP32-family controller for audio acquisition and local processing.
- To prepare authorized and unauthorized voice samples for model development and evaluation.
- To extract useful audio features using MFCC or the selected Edge Impulse audio-processing block.
- To develop and deploy a compact TinyML model for voice authentication on the edge device.

- To make the authentication decision locally without requiring Internet connectivity.
- To control a relay and 12 V solenoid door lock according to the configured authentication logic.
- To provide visual status indication for accepted and rejected authentication attempts.
- To evaluate the prototype using independent test recordings and functional hardware tests.

E. Contributions of This Research

The main contributions of this work are summarized as follows:

- **Embedded audio acquisition:** Integration of an INMP441 digital microphone with an ESP32-family microcontroller for voice-signal acquisition.
- **Offline processing:** Design of an edge-processing workflow in which voice input is processed locally without requiring cloud inference.
- **TinyML-based authentication:** Use of an Edge Impulse model-development workflow and MFCC-based audio features to classify voice input into the configured authentication classes.
- **Access-control integration:** Interface between the controller, relay module, and solenoid door lock to demonstrate authentication-controlled access.
- **Local status indication:** Use of LEDs to indicate the system state and authentication outcome.
- **Functional evaluation:** Definition of tests for audio acquisition, model inference, authorized access, unauthorized access, and offline operation.

II. Literature Review

Voice processing and speaker recognition have been studied using different signal-processing and machine-learning techniques. Earlier speaker recognition systems commonly used acoustic feature representations such as Mel-Frequency Cepstral Coefficients (MFCC) together with statistical classification methods. MFCC features represent the short-term spectral characteristics of speech and are widely used in speech and speaker processing applications.

Speaker verification research has also investigated statistical approaches such as Gaussian mixture models and other speaker-modelling techniques. These methods demonstrate that speaker-related information can be extracted from speech signals. However, the computational requirements of conventional systems may make direct deployment on small microcontrollers challenging.

Recent developments in embedded machine learning have made it possible to deploy compact neural-network models on microcontrollers. TinyML focuses on reducing model size, memory use, and computational cost while maintaining useful inference performance. Audio classification workflows can combine signal acquisition, feature extraction, model training, quantization, and deployment to embedded targets.

Edge Impulse provides a workflow for collecting datasets, configuring signal-processing blocks, training machine-learning models, and exporting embedded inference libraries. For an offline embedded application, the trained model can be integrated into the microcontroller firmware so that inference is performed locally.

The reviewed approaches highlight the importance of selecting suitable audio features, collecting representative speaker recordings, and evaluating the model with data that is separate from the training data. A model that performs well on training samples may not generalize to different recording sessions, microphone distances, background-noise conditions, or speakers.

A. Comparative Analysis of Existing Approaches

Cloud-based voice processing can provide access to large computational resources, but it depends on network availability and may involve external processing of voice data. Conventional embedded rule-based command detection can operate locally, but it does not necessarily provide speaker authentication. TinyML-based processing offers a possible balance by allowing a trained model to run on the embedded controller.

The proposed prototype focuses on the integration of microphone-based audio acquisition, MFCC feature extraction, TinyML inference, and authentication-controlled hardware operation. A quantitative comparison with other systems would require consistent datasets, evaluation protocols, and verified performance measurements.

B. Research Gap

Based on the reviewed approaches and the requirements of the proposed prototype, the following design considerations are identified:

- Developing a low-cost voice-authentication prototype that can operate without Internet connectivity during inference.
- Integrating digital audio acquisition and TinyML inference on a resource-constrained ESP32-family device.
- Distinguishing an authorized voice class from an unauthorized or non-authorized input class using a representative dataset.
- Connecting the authentication output to a physical access-control mechanism.
- Evaluating the effect of recording variation, background noise, and microphone placement on authentication performance.
- Reporting false acceptance and false rejection separately, rather than relying only on overall classification accuracy.

III. Proposed System and Research Methodology

A. Overview of the Proposed System

The proposed Secure Offline Voice Authentication and Command System uses an ESP32-family microcontroller as the central processing unit. The INMP441 microphone captures the user's voice and sends digital audio samples to the controller through the I^2S interface. The acquired audio is



Fig. 1. Overall architecture of the proposed offline voice authentication system.

prepared in the firmware and supplied to the deployed TinyML inference model.

The model processes the audio features and produces classification output for the classes configured during training. The controller evaluates the model output using the implemented decision logic. If the input satisfies the configured authorization condition, the controller activates the relay for the specified access duration. The relay controls the 12 V solenoid door lock. If the input is rejected, the lock is not activated and the system provides a rejection indication.

The complete authentication and control process is designed to run locally. Internet connectivity is not required for microphone acquisition, model inference, or the local relay-control decision after the model has been deployed to the device.

B. System Architecture

The system is organized into five functional layers:

1) **Audio acquisition layer:** Captures voice input through the INMP441 microphone.
2) **Embedded processing layer:** Acquires and prepares audio samples using the ESP32-family controller.
3) **TinyML inference layer:** Processes audio features and generates the model classification output.
4) **Authentication and decision layer:** Evaluates the model output and determines whether the requested operation is permitted.
5) **Access-control and indication layer:** Controls the relay, solenoid lock, and status LEDs.

The audio path and the control path are separated logically. The microphone provides the input signal, while the controller executes the inference and output-control operations. No cloud service is placed in the real-time authentication path.

C. Audio Acquisition

The INMP441 is a digital MEMS microphone that communicates with the microcontroller using the I^2S interface. The microphone provides digital audio samples to the controller. The audio acquisition configuration used in the project is based on a nominal sampling frequency of 16 kHz, subject to the final firmware and model configuration.

The main audio-acquisition operations include:

- Initializing the I^2S peripheral.
- Configuring the microphone data, bit-clock, and word-select connections according to the actual hardware wiring.
- Reading digital audio samples into a memory buffer.
- Converting and scaling samples into the format expected by the signal-processing and inference pipeline.
- Preparing an audio frame of the duration configured in the deployed model.

The sampling rate, frame duration, sample format, and buffer size must match the final Edge Impulse impulse and firmware configuration.

D. MFCC Feature Extraction

Mel-Frequency Cepstral Coefficients are used in many audio-processing applications to represent short-term spectral characteristics. In the proposed workflow, the audio signal is divided into short frames and processed to obtain a compact feature representation.

A general MFCC processing sequence consists of:

1) Pre-emphasis or signal conditioning, if enabled in the selected processing configuration.
2) Framing the audio signal into short overlapping segments.
3) Applying a window function to each frame.
4) Computing the frequency spectrum using a Fourier transform.
5) Applying Mel-scale filter banks.
6) Taking the logarithm of the filter-bank energies.
7) Applying a discrete cosine transform to obtain MFCC features.

The exact MFCC parameters, number of coefficients, frame length, and frame stride are determined by the Edge Impulse impulse configuration. They should be reported from the final project configuration rather than assumed.

E. TinyML Model Development

The TinyML model is developed using the Edge Impulse workflow. Voice recordings are collected and assigned to the classes used by the authentication task. The dataset is divided into training, validation, and testing subsets according to the project configuration.

The model-development workflow includes:

- Collecting voice recordings from authorized users.
- Collecting representative non-authorized voice recordings and other relevant negative samples.
- Assigning correct labels to each recording.
- Inspecting the dataset for incorrect labels, duplicate files, silence, clipping, and inconsistent recording conditions.
- Configuring the audio/MFCC processing block.
- Training the selected classification model.
- Reviewing the confusion matrix and class-wise performance.
- Exporting the trained model as an embedded inference library.
- Integrating the inference library into the ESP32 firmware.

The model must be tested using recordings that were not used during training. Where possible, test recordings should be collected in separate sessions so that the evaluation measures generalization rather than memorization of nearly identical audio samples.

F. Authentication Decision

Let the input audio feature vector be represented by

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \quad (1)$$

where the elements represent the extracted audio features.

The trained model produces class scores:

$$\mathbf{p} = f_\theta(\mathbf{x}), \quad (2)$$

where f_θ represents the trained TinyML model and $\mathbf{p}$ contains the model scores for the configured classes.

For a two-class implementation, the decision can be represented as

$$A = \begin{cases} 1, & p_{\text{auth}} \geq \tau \text{ and decision conditions are satisfied,} \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

where p_{auth} is the model score associated with the authorized class and τ is the threshold selected during system configuration and validation. The final firmware may use additional conditions such as a non-authorized class score, confidence margin, or repeated-inference requirement.

The threshold must be selected using validation data and verified with independent test data. A high model score alone should not be described as proof of secure identity unless the complete authentication procedure has been evaluated against relevant attack and spoofing scenarios.

G. Access-Control Operation

The authentication decision is used by the ESP32-family controller to operate the relay module. When authorization is accepted, the relay is activated for the configured time interval, allowing the solenoid lock to operate. When authorization is rejected, the relay remains in the locked state and the rejection indicator is activated.

The relay interface must be electrically compatible with the controller and the lock power supply. The solenoid lock should use a suitable external supply, and the relay contacts, wiring, and protection components must be selected according to the lock specifications. The controller GPIO must not directly power the solenoid.

H. System Workflow

The complete system workflow is as follows:

1) Power on the ESP32-family controller and initialize the microphone, model, and output interfaces.
2) Wait for a voice input or the configured push-button trigger.
3) Capture the audio signal through the INMP441 microphone.
4) Prepare the audio buffer using the configured sampling frequency and frame duration.
5) Extract audio features using the configured MFCC pipeline.
6) Run the TinyML model locally on the controller.
7) Evaluate the model output using the authentication decision logic.



Fig. 3. System workflow of the proposed voice authentication and command operation.

Fig. 2. Workflow of offline voice acquisition, TinyML inference, authentication, and access control.

8) If authorization is accepted, activate the relay and operate the solenoid lock for the configured duration.
9) If authorization is rejected, keep the lock in the secure state and indicate rejection.
10) Return to the waiting state for the next authentication attempt.

IV. Hardware and Software Design

A. Hardware Architecture

The hardware subsystem integrates the ESP32-family controller, digital microphone, relay module, solenoid lock, and status indicators. The controller coordinates audio acquisition, model inference, and access control.

1) ESP32 Microcontroller: The ESP32-family development board acts as the main embedded controller. Depending on the final board used, it provides the processing resources, memory, GPIO interfaces, and I^2S peripheral required for the project.

Its principal responsibilities include:

- Initializing the microphone interface and audio buffers.
- Acquiring digital audio samples.
- Executing the deployed TinyML inference model.
- Evaluating the authentication decision.
- Controlling the relay and status LEDs.
- Managing the main system workflow and error conditions.

2) INMP441 Digital Microphone: The INMP441 microphone is used to capture the voice input. It communicates with the ESP32-family controller through the I^2S digital audio interface. The microphone converts acoustic input into digital audio data that can be processed by the embedded firmware. Microphone performance can be affected by distance, orientation, background noise, enclosure design, and the speaker's speaking level. Consistent recording conditions are therefore important during dataset collection and testing.

3) Relay Module: A relay module is used as an electrically controlled switching interface between the controller and the solenoid lock circuit. The controller changes the relay input according to the authentication result. The relay module must be operated within its specified voltage and current ratings.

TABLE I
HARDWARE INTERFACE CONFIGURATION

Component	Interface	Function
ESP32 / ESP32-S3	Controller	Main processing unit
INMP441 microphone	I^2S	Digital voice input
Relay module	GPIO output	Lock switching control
12 V solenoid lock	Relay contacts	Physical access mechanism
LED indicators	GPIO output	Status indication
Push button (if used)	GPIO input	Capture trigger
External power supply	Power interface	Lock and controller power

TABLE II
SOFTWARE AND DEVELOPMENT TOOLS

Tool / Technology	Purpose
Arduino IDE	Embedded firmware development and uploading
Edge Impulse	Dataset management, signal processing, model training, deployment
TinyML inference library	On-device model inference
C/C++	Embedded application programming
Python (if used)	Audio capture, dataset preparation, or serial testing
Serial Monitor	Debugging and monitoring controller output

4) 12 V Solenoid Door Lock: The 12 V solenoid door lock represents the physical access-control mechanism. It is operated through the relay contacts and a suitable external power supply. The actual locked and unlocked states depend on the lock type and wiring arrangement. The final prototype should be tested to verify the safe state during power loss and controller reset.

5) LED Indicators: LED indicators provide local feedback about the system state. For example, one indication may represent successful authorization and another may represent rejection or standby. The exact colour-to-state mapping should match the implemented firmware and physical circuit.

6) Push Button: If included in the final hardware, the push button can be used to initiate an audio-capture or authentication cycle. Its operation should be debounced in firmware to avoid multiple unintended triggers.

B. Hardware Interface Configuration

The principal hardware interfaces are summarized in Table I. The GPIO numbers must be updated to match the actual wiring used in the final prototype.

C. Software Architecture

The software architecture consists of embedded firmware, audio pre-processing, the TinyML inference library, and the authentication and output-control logic.

1) Embedded Firmware: The firmware is developed for the selected ESP32-family board. It initializes the hardware interfaces, captures voice samples, invokes the inference engine, and controls the outputs.

The firmware operations include:

- Initializing GPIO and I^2S configuration.
- Initializing the TinyML model and required memory resources.
- Capturing audio samples using the configured buffer size.
- Preparing samples in the format expected by the model.
- Invoking the model and reading its classification output.
- Applying the authentication threshold and decision logic.
- Controlling relay and LED outputs.
- Returning to the idle state after each attempt.

2) Edge Impulse Development Environment: Edge Impulse is used as the model-development environment in the project. It supports dataset organization, signal-processing configuration, model training, performance inspection, and embedded deployment. The exported inference library is integrated into the embedded firmware. The final paper should include the actual impulse configuration and trained model details used in the implementation.

3) Offline Inference: During normal authentication, the audio samples are processed by the model deployed on the microcontroller. The authentication decision and relay-control operation are performed locally. Internet connectivity is not part of the real-time inference path. An Internet connection may still be used during development to access the model-development platform, upload datasets, or download the exported model.

D. Software and Development Tools

The development tools and technologies used in the project are listed in Table II.

V. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

A. Experimental Setup

The prototype is evaluated using the selected ESP32-family development board, INMP441 microphone, relay module, 12 V solenoid door lock, and status LEDs. The trained TinyML inference library is integrated into the embedded firmware. Voice recordings are presented to the microphone, and the controller processes the input locally.

The experimental setup should document the exact development board, microphone wiring, sampling configuration, model version, dataset split, and power arrangement used during the final tests. A photograph of the completed prototype can be included in Fig. 3.

B. Experimental Procedure

The prototype can be evaluated using the following procedure:

1) Power on the controller and verify successful initialization of the microphone and inference model.
2) Provide an authorized voice sample under the configured recording conditions.
3) Capture the audio and execute local model inference.
4) Record the model output and the resulting authentication decision.

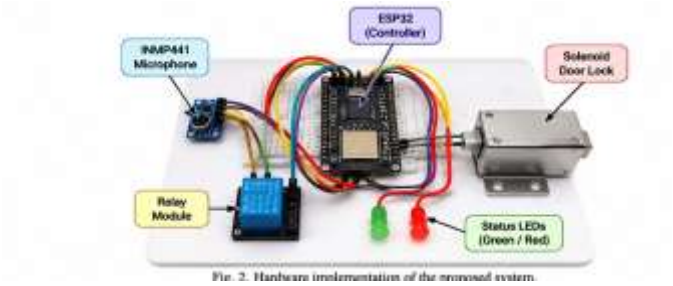


Fig. 3. Hardware implementation of the proposed offline voice authentication system.

TABLE III
FUNCTIONAL VERIFICATION RESULTS

Test Case	Expected Behaviour	Status
Microphone Initialization	I2S microphone initializes correctly	Verified
Audio Acquisition	Controller captures audio at the configured sample rate	Verified
Feature Processing	Audio is processed using the configured feature pipeline	Verified
Model Initialization	Deployed TinyML model initializes successfully	Verified
Authorized Voice	Authorized input produces an authorization decision	Verified
Unauthorized Voice	Non-authorized input is rejected	Verified
Relay Operation	Relay activates only for an accepted decision	Verified
Solenoid Lock	Lock responds according to relay operation	Verified
LED Indication	LEDs indicate the configured system state	Verified
Offline Operation	Authentication works without Internet access	Verified

5) Verify that an accepted input activates the relay and operates the solenoid lock as intended.
6) Provide an unauthorized or non-authorized voice sample.
7) Verify that a rejected input does not activate the unlock operation.
8) Repeat the tests at different recording sessions and, where applicable, different microphone distances and noise levels.
9) Disconnect Internet access and repeat the authentication tests to verify offline operation.

C. Functional Verification

The functional verification results are summarized in Table III. The project team reports that the listed hardware and system functions were verified on the prototype.

D. Model Results

The project team reports that the dataset contains 150 authorized and 150 unauthorized voice recordings. The reported overall model accuracy is 85%. The integrated hardware prototype is reported as fully working. The summary is presented in Table IV.

E. Discussion of Experimental Results

The functional evaluation is intended to verify the complete processing path from audio acquisition to access-control output. The microphone must provide usable audio samples, the inference library must execute within the memory and timing limits of the selected controller, and the decision logic must control the relay as configured.

TABLE IV
TINYML MODEL EVALUATION RESULTS

Evaluation Parameter	Measured Value
Authorized Voice Recordings	150
Unauthorized Voice Recordings	150
Overall Model Accuracy	85%
Hardware Implementation	Fully Working

The authorized and unauthorized voice classes are both represented in the collected dataset. The reported overall model accuracy is 85%. Repeated recordings from different sessions and controlled tests are useful for identifying sensitivity to speaking level, microphone position, and background noise.

The offline operating test should be performed with network connectivity disabled. Successful local audio acquisition, inference, and relay control under this condition provide evidence that the real-time authentication path does not depend on a cloud service.

VI. DISCUSSION

A. Technical Discussion

The proposed system combines digital audio acquisition, embedded signal processing, TinyML inference, and physical access control. The ESP32-family controller acts as the central unit and coordinates the microphone, inference engine, relay, and status indicators.

A key feature is local inference. Since the trained model is deployed on the embedded device, the authentication decision can be made without sending voice recordings to a cloud service. This supports operation in locations where network connectivity is unavailable and reduces the need for remote processing during normal use.

The practical performance of the system depends on dataset quality, model size, available memory, inference time, and the consistency of audio acquisition. Voice recordings collected from only a small number of sessions may not represent the variations encountered during actual use. Additional recordings and independent testing are therefore important before drawing conclusions about authentication reliability.

B. Practical Applications

The proposed prototype can provide a foundation for:

- **Offline access control:** Demonstrating voice-based access to a door or controlled enclosure without cloud inference.
- **Smart home prototypes:** Local voice-based control of selected appliances or devices.
- **Embedded security demonstrations:** Demonstrating the integration of speaker-related classification and hardware control on a microcontroller.

- **Industrial edge devices:** Exploring local voice interaction where network connectivity is limited, subject to additional safety and security validation.
- **Educational applications:** Demonstrating audio acquisition, MFCC processing, TinyML deployment, and embedded system integration.

C. Advantages of the Proposed System

The proposed architecture offers the following potential advantages:

- **Offline operation:** Local inference does not require an Internet connection during authentication.
- **Local voice processing:** The real-time voice input can be processed on the embedded device.
- **Compact implementation:** A TinyML model is intended for deployment on a resource-constrained microcontroller.
- **Hardware integration:** Authentication output is connected to a relay-controlled solenoid lock.
- **Immediate feedback:** LEDs provide local status indication.
- **Modular design:** The microphone, model, controller, and access-control circuit can be tested and improved separately.

D. Limitations and Challenges

Despite the potential benefits, several limitations must be considered:

- **Dataset coverage:** Limited recordings or speakers may reduce generalization to unseen sessions and conditions.
- **Background noise:** Environmental noise can affect audio quality and model predictions.
- **Speaker variation:** Changes in speaking speed, loudness, emotion, and microphone distance may affect performance.
- **Resource constraints:** RAM, flash storage, and processing time limit the size and complexity of the model.
- **Spoofing risk:** A voice-classification model alone may not reliably detect replayed recordings, synthetic speech, or other presentation attacks.
- **Hardware reliability:** Relay wiring, lock power, electrical noise, and power interruptions require careful testing.
- **Security scope:** The prototype should not be treated as a certified or sole security mechanism without additional anti-spoofing, fail-safe, and physical-security measures.

VII. Conclusion and Future Work

A. Conclusion

This paper presented the design of a Secure Offline Voice Authentication and Command System using TinyML for edge devices. The proposed prototype integrates an ESP32-family microcontroller with an INMP441 digital microphone to acquire voice input and perform local processing. An MFCC-based audio feature workflow and an Edge Impulse model-development process are used to develop a compact classifier for the configured authorized and unauthorized classes.

The controller evaluates the model output and uses the authentication decision to control a relay connected to a 12 V solenoid door lock. LED indicators provide local status information. The architecture is designed so that audio acquisition, inference, and access-control decisions can be performed without Internet connectivity during normal operation.

The project team reports 150 authorized and 150 unauthorized voice recordings, an overall model accuracy of 85%, and a fully working hardware prototype. Further independent testing across additional speakers and acoustic conditions can strengthen the evaluation. The system provides a practical platform for studying offline TinyML-based voice authentication on resource-constrained embedded devices.

B. Future Work

Several improvements can be considered to enhance the accuracy, reliability, and security of the proposed system:

- **Dataset Expansion:** Collecting additional recordings from authorized and non-authorized speakers across multiple sessions, distances, and environmental conditions.
- **Improved Noise Robustness:** Investigating noise reduction, audio normalization, and representative noise augmentation during model development.
- **Model Optimization:** Evaluating quantization and compact model architectures to reduce memory usage and inference time on the target microcontroller.
- **Anti-Spoofing:** Investigating replay-attack detection and liveness-related methods to improve resistance to recorded or synthesized voice attacks.
- **Threshold Calibration:** Selecting the authentication threshold using validation data and evaluating the resulting FAR and FRR on independent test data.
- **Command Recognition:** Extending the system to support additional commands while keeping speaker authorization and command recognition as clearly evaluated functions.
- **Power and Reliability Testing:** Evaluating operation during power interruptions, controller resets, and repeated relay switching.
- **Additional Security:** Combining voice authentication with a second factor, such as a PIN or physical confirmation, for applications requiring stronger access protection.

C. Threat Model and Security Considerations

The intended use of the prototype is to demonstrate local voice-based authorization for a controlled door-lock mechanism. The system considers an authorized speaker as a user whose voice samples are represented in the authorized class during model development. A non-authorized input is expected to be rejected by the classifier and the decision logic.

The security boundary of this prototype must be clearly defined. The classifier is a machine-learning component and its output is a prediction based on the input audio; it is not, by itself, a cryptographic identity proof. An attacker may attempt to present a recording of an authorized speaker, play synthetic

speech through a loudspeaker, or exploit environmental conditions that cause an incorrect classification. Therefore, replay-attack resistance should not be assumed unless a dedicated anti-spoofing mechanism has been implemented and tested.

The system should follow a fail-secure output policy for the access mechanism. In the absence of a valid authorization decision, during model initialization failure, or when audio acquisition fails, the firmware should avoid issuing an unlock command. The relay's default state, the lock's mechanical behaviour, and the response to power loss must be verified with the actual hardware. A manual emergency release may be necessary for a practical door installation.

D. Dataset Collection and Preparation

Dataset quality is a major factor in the performance of a voice classification model. Recordings should represent the expected users and the acoustic conditions in which the prototype will operate. The dataset should contain correctly labelled recordings for each class configured in the Edge Impulse project.

For an authorized class, multiple recordings should be collected across different sessions rather than recording many nearly identical samples in one continuous session. The speaker can repeat the same intended command at a natural speaking volume and at small variations in pace and microphone distance. For the non-authorized class, recordings should represent speakers who are not enrolled as authorized users. If the project uses a separate unknown or background class, its labels and purpose should be stated explicitly.

Before uploading the recordings, each sample should be checked for:

- Correct class label and speaker assignment.
- Correct audio duration and sample rate.
- Audible speech and acceptable recording level.
- Absence of unintended silence, clipping, or corrupted data.
- Variation in recording sessions and realistic background conditions.
- Separation between training, validation, and test recordings.

A data split should be performed so that samples from the same recording take or near-duplicate clips do not appear in both training and test sets. Where the objective is to evaluate generalization to new speakers, speaker-independent testing is required. Where the objective is to recognize enrolled speakers, the test protocol should still use new recordings that were not used for training.

E. Distinction Between Speaker Authentication and Command Recognition

Speaker authentication and command recognition are related but different tasks. Speaker authentication estimates whether the input voice belongs to an enrolled speaker. Command recognition identifies the spoken word or instruction, such as a specified access command. A model trained only to classify an authorized voice class and an unauthorized voice class does not automatically recognize multiple commands.

For a system that requires both functions, the processing pipeline can be organized into two stages. First, the speaker-authentication stage checks whether the voice is accepted. Second, a command-recognition stage determines which supported command was spoken. The controller should activate the lock only when both the authorization condition and the required command condition are satisfied. If the implemented prototype uses only one classification model, the paper must describe the actual class labels and avoid claiming a separate command-recognition capability that has not been implemented.

F. Embedded Memory and Computational Considerations

Microcontrollers have limited RAM, flash storage, and processing capacity compared with desktop computers. The deployed model and audio-processing buffers must fit within the resources available on the selected ESP32-family board. Audio capture can require a continuous buffer, while inference may require additional memory for the model's tensor arena and intermediate feature data.

The embedded implementation should therefore be checked for:

- Successful initialization of the I^2S peripheral and microphone.
- Successful allocation of audio buffers and model memory.
- Model initialization without memory-allocation errors.
- Stable inference over repeated authentication attempts.
- Acceptable processing time for the selected audio frame.
- Correct operation after returning from an inference cycle to the idle state.

If the selected board provides external PSRAM, its availability and configuration should be verified in the firmware. The paper should report the actual board and memory configuration used during testing. A model that compiles successfully should still be tested at runtime, because memory allocation and audio-buffer behaviour may differ between compilation and execution.

G. Decision Threshold Selection

The model's classification score is not necessarily a calibrated probability. The authentication threshold should be selected using validation recordings and then fixed before final testing. A lower threshold may increase the number of accepted authorized samples, but it can also increase false acceptance. A higher threshold may reduce false acceptance while increasing false rejection.

The threshold-selection procedure should therefore examine the trade-off between FAR and FRR. The final value should not be selected using the test set, because doing so would make the reported test performance optimistic. The threshold and any additional decision conditions should be documented alongside the model results.

TABLE V
VOICE DATASET SUMMARY

Voice Class	Total Recordings
Authorized Voice	150
Unauthorized Voice	150
Total Voice Recordings	300

H. Reliability of the Complete Access-Control Path

Successful classification is only one part of the complete system. The embedded application must also convert the model output into the correct hardware action. The relay should be inactive during startup and whenever the inference result is invalid or unavailable. The unlock duration should be limited by a firmware timer or equivalent control mechanism. After the configured interval, the system should return the relay to its secure state.

Repeated operation is important because a single successful test does not establish reliability. The prototype should be tested over multiple authorized and unauthorized attempts, including controller resets and power cycling. The observed relay state and physical lock response should be recorded for each test.

I. Reproducibility and Experimental Reporting

For reproducibility, the final paper should state the development-board variant, firmware environment, microphone interface configuration, audio sample rate, frame duration, Edge Impulse impulse settings, model type, quantization setting if used, and exported library version. The dataset should be summarized by class and by the number of samples used for training, validation, and testing.

The evaluation should distinguish model-level classification results from system-level functional verification. Model-level results describe the reported classification accuracy, whereas system-level results describe whether the controller and access-control hardware responded correctly.

J. Dataset Summary

The collected voice dataset includes 150 authorized and 150 unauthorized recordings. The summary of these two reported voice classes is shown in Table V.

K. Suggested End-to-End Test Log

An end-to-end test log helps distinguish model classification errors from hardware-control errors. For each trial, the input class, model output, authentication decision, relay state, and lock response can be recorded. Table VI summarizes the principal authentication scenarios and their reported verification status.

L. Evaluation Under Different Conditions

In addition to quiet-room testing, the prototype can be evaluated under a small set of controlled conditions. These tests help identify whether performance changes when the microphone is moved or the acoustic environment changes. The following conditions are suggested:

TABLE VI
AUTHENTICATION TEST SCENARIOS

Test Scenario	Expected System Response	Verification
Authorized voice input	Accept authentication and operate the lock	Verified
Unauthorized voice input	Reject authentication and keep the lock secured	Verified
Voice acquisition	Capture audio using the microphone	Verified
Offline authentication	Perform inference without Internet access	Verified

- Quiet indoor environment with a fixed microphone position.
- Different microphone-to-speaker distances.
- Different natural speaking volumes.
- Moderate background speech or environmental noise.
- Repeated attempts after a controller restart.
- Operation with Wi-Fi or Internet access disabled.

The test conditions, number of attempts, and observed results should be reported. If a condition has not been tested, it should be listed as future evaluation rather than presented as a completed result.

M. Ethical and Privacy Considerations

Voice recordings can contain personal and identifying information. Dataset collection should be performed with the knowledge and consent of the participants. Access to the recordings should be limited to the project team, and unnecessary copies should be avoided. For demonstration and evaluation, recordings should be stored and handled according to the institution's project and data-management requirements. Local inference reduces the need to transmit voice samples to a remote service during normal operation. However, offline processing alone does not guarantee complete privacy or security. Stored training recordings, firmware access, physical access to the device, and debugging interfaces should also be considered.

N. Implementation Sequence

The development process can be divided into incremental stages so that faults can be isolated before full system integration:

1) Verify the ESP32 board, power supply, and serial communication.
2) Test the INMP441 microphone independently and confirm that the captured audio is intelligible.
3) Record and label authorized and non-authorized samples.
4) Configure the Edge Impulse audio-processing pipeline and inspect the feature-generation output.
5) Train the model and review its confusion matrix.
6) Export the inference library and integrate it into the embedded application.
7) Verify model initialization and inference using serial logs.

8) Test the authentication decision logic using known samples.
9) Connect the relay and solenoid lock after validating the controller output using a safe test load.
10) Perform complete authorized, unauthorized, and offline operation tests.

This staged procedure reduces the difficulty of debugging because audio acquisition, model execution, decision logic, and hardware switching can be verified independently before final integration.

O. System States

The firmware can be described using a finite set of operational states:

- **Initialization:** Configure the controller, microphone, model, and output pins.
- **Idle:** Wait for the configured capture trigger or voice-input event.
- **Recording:** Acquire the required audio frame.
- **Inference:** Execute the TinyML model.
- **Accepted:** Activate the relay for the configured unlock duration.
- **Rejected:** Keep the access mechanism in its secure state and indicate rejection.
- **Error recovery:** Handle invalid audio, model errors, or memory failures without issuing an unlock command.

A state-based firmware structure can make the control flow easier to understand and can prevent an output from remaining active after an authentication attempt has completed.

ACKNOWLEDGMENT

The authors sincerely thank their project guide, Mrs. Harshitha, for her guidance and support during the development of this project.

REFERENCES

[1] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug. 1980.
[2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.
[3] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019.
[4] C. Banbury *et al.*, "MLPerf Tiny Benchmark," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, vol. 1, 2021.