ECE 445
Spring 2026

Project #80: Edge-AI based audio classifier

Team Members:
Ahaan Joishy (ajoishy2)
Kavin Manivasagam (kmani4)
Om Dhingra (omd4)

TA: Weijie Liang

1. Introduction

1.1 Problem

Most embedded audio-based systems rely on a single microphone and simple threshold-based logic to detect sounds, such as triggering when the signal amplitude exceeds a predefined level. While this approach is computationally inexpensive and easy to implement, it is highly sensitive to environmental noise and background interference. In real-world environments, ambient sounds such as wind, traffic, conversations, or mechanical noise can easily cause false triggers or mask important audio events. Additionally, amplitude-based detection cannot differentiate between different types of sounds that share similar loudness characteristics. As a result, these systems lack robustness and are unsuitable for applications requiring reliable classification or contextual awareness.

Another major limitation of single-sensor systems is their inability to determine the direction of a sound source. Without spatial information, the system cannot distinguish whether a detected sound originates directly in front of the device or from a distant or irrelevant direction. This prevents the implementation of localization-based filtering, beamforming, or directional response mechanisms. In applications such as environmental monitoring, wildlife observation, or smart sensing platforms, the ability to localize a sound source is critical for extracting meaningful information from complex acoustic environments.

Cloud-based audio processing platforms attempt to address these limitations by leveraging powerful remote servers to perform advanced signal processing and machine learning inference. These systems can execute computationally intensive algorithms such as convolutional neural networks, spectral analysis, and source separation with high accuracy. However, offloading data to the cloud has several problems. First, network latency can significantly delay response times, making real-time operation difficult or impossible in bandwidth-limited or offline environments. Second, transmitting raw audio data raises privacy and security concerns, particularly in applications deployed in public or residential spaces. Third, continuous wireless communication increases power consumption, reducing battery life and limiting portability.

Therefore, there is a clear need for a low-power embedded system capable of performing real-time sound classification and sound source localization entirely on-device. Such a system must operate under strict memory, processing, and energy constraints while maintaining acceptable accuracy and latency. By integrating efficient signal processing techniques, lightweight machine learning models, and optimized firmware on a microcontroller platform such as the ESP32, it is possible to create a self-contained acoustic sensing system. This approach eliminates reliance on cloud infrastructure, reduces latency, preserves user privacy, and enables deployment in remote or resource-constrained environments. Developing such a system represents a significant improvement over traditional threshold-based detection methods and bridges the gap between simple embedded sensing and advanced intelligent audio analytics.

Solution

This project proposes an Edge-AI embedded system capable of performing real-time sound classification and sound source localization using multiple digital microphones and a neural network deployed on a low-power microcontroller. The system extracts spectral features from incoming audio signals, estimates the direction of arrival of the sound source, and classifies the sound into predefined categories. The results are displayed using a directional LED array or small display, allowing the user to visually identify both the type and location of the detected sound. All processing is performed locally without relying on cloud services, demonstrating the feasibility of embedded machine learning under strict power and memory constraints.

1.2 Visual Aid

1.3 High Level Requirements

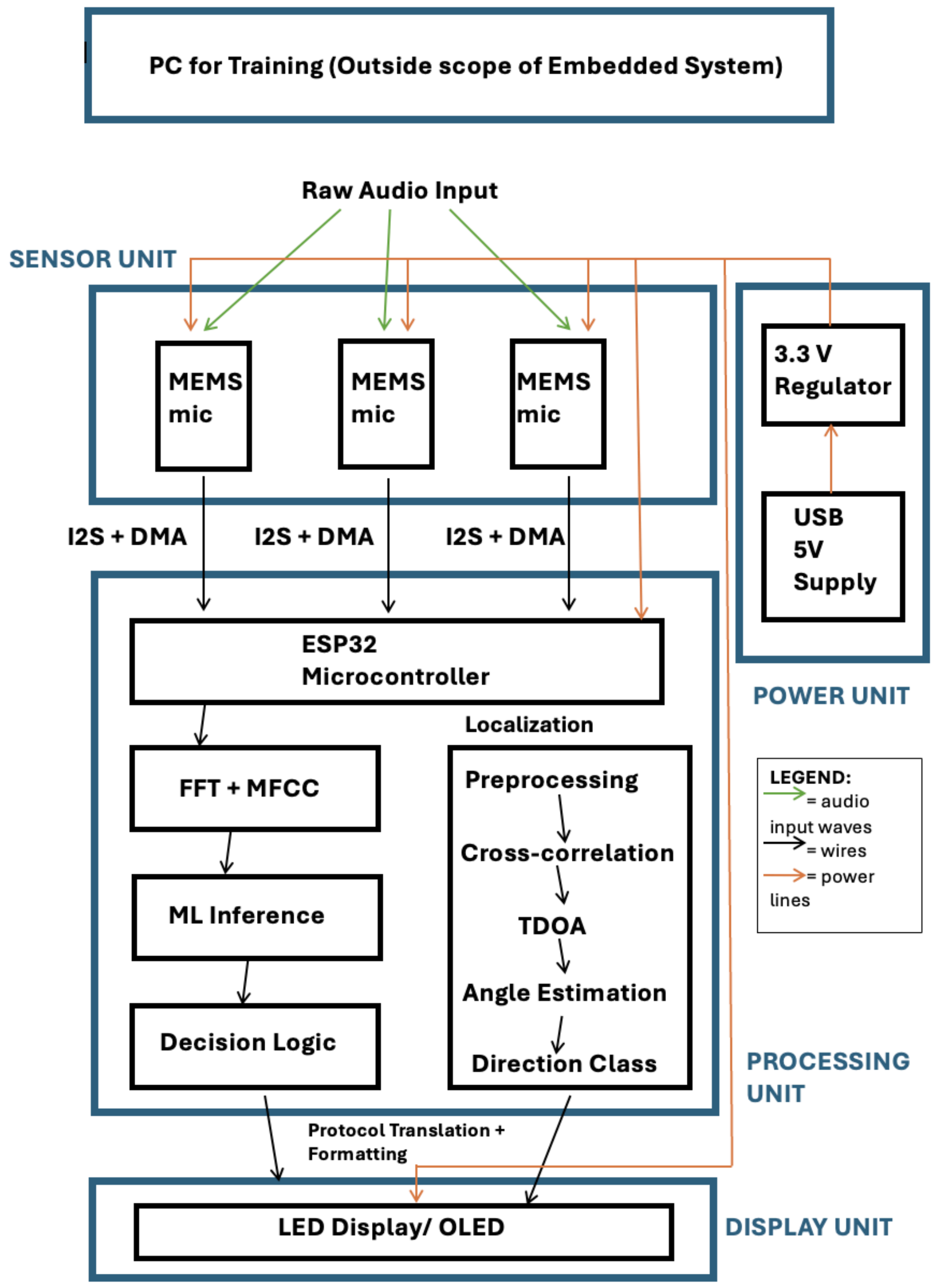
- The system shall correctly classify at least three distinct bird species vocalizations with a minimum accuracy of 85% on a recorded test dataset collected in varied environmental conditions.

- The system shall perform real-time sound localization and classification with an end-to-end latency of less than 100 milliseconds from sound detection to visual display output.

- The device shall operate entirely on-device without cloud computation while maintaining stable operation under a maximum average current consumption of 150 mA from a 5V USB power source.

2.1 Design

2.1.1 Block Diagram



2.2 Subsystem Overview

2.2.1 Subsystem 1: Sensor Subsystem:

The Sensor Subsystem is responsible for capturing synchronized audio input from the environment and delivering digital audio streams to the Processing Subsystem for feature extraction and classification. It serves as the entry point for all acoustic data in the system, and its quality and timing accuracy directly determine the performance of both the ML classification pipeline and the TDOA-based localization algorithm downstream.

Component Selection

We selected the ICS-43434 digital MEMS microphone for this subsystem. It outputs audio directly over an I2S interface, eliminating the need for analog amplification, anti-aliasing filters, or ADC circuitry that would otherwise add component count, board complexity, and potential noise sources. The ICS-43434 operates on a 3.3V supply, making it directly compatible with our power subsystem without level shifting. It supports a sample rate of 16 kHz, which is sufficient to capture the frequency content of bird vocalizations, which typically fall below 8 kHz, satisfying the Nyquist criterion. Two microphones are placed at a known fixed separation distance on the PCB. This spacing is critical for the TDOA localization algorithm in the Processing Subsystem, the physical distance between microphones determines the maximum resolvable time delay and therefore the angular resolution of the system.

Audio data is transferred from both microphones to the ESP32-S3 via I2S with DMA. Using DMA offloads the transfer entirely from the CPU, ensuring that audio capture does not block the processing core during FFT, MFCC extraction, or ML inference. The two microphones share the same I2S clock and word select lines to ensure synchronized sampling, which is essential for cross-correlation accuracy in the localization pipeline. Without synchronized capture, phase differences between microphone signals would be corrupted, making TDOA estimation unreliable.

Relationship to Other Subsystems

The Sensor Subsystem feeds raw 16 kHz PCM audio buffers directly into the Processing Subsystem via DMA memory buffers. It has no direct interface with the Display Subsystem or Power Subsystem beyond receiving a clean 3.3V supply. The quality of the audio captured here directly bounds the maximum achievable classification accuracy and localization precision of the entire system.

Requirement	**Verification**
● Both microphones must sample audio synchronously at 16 kHz with no clock drift between channels.	● Configure both microphones on a shared I2S clock. Capture 1 second of audio from both channels simultaneously while playing a 1 kHz tone from a known source. Read both DMA buffers from the ESP32 over serial. Confirm that the waveforms in both buffers are phase-coherent and that the sample count in each buffer is identical (16,000 samples ± 0).
● Each microphone must successfully deliver audio data to the ESP32 via I2S DMA with no buffer overruns during continuous operation.	● Run continuous I2S DMA capture for 60 seconds. Monitor the ESP32 serial output for DMA overflow or error flags. Confirm that zero buffer overrun events are logged over the full 60-second period.
● The microphone placement must produce a measurable and consistent TDOA between the two channels when a sound source is positioned at a known off-center angle.	● Place a speaker at a 45-degree angle relative to the microphone axis at a distance of 1 meter. Capture a short broadband audio burst (e.g., a clap or impulse) from both microphones. Compute the cross-correlation of the two channels in Python on a PC using the same algorithm to be deployed on-device. Confirm that the peak cross-correlation lag corresponds to the expected

	TDOA of d·sin(45°)/c, where d is the microphone separation and c is the speed of sound, within a tolerance of ±1 sample at 16 kHz.

2.2.2 Subsystem 2: Processing Subsystem:

The Processing Subsystem is the main computational core of the system. It receives digitized audio signals from the Sensor Subsystem, performs digital signal processing (DSP), ML inference on MFCC features, sound source localization, and then sends the results to the Output/Display Subsystem for visualization. It essentially is responsible for classifying the type of sound using an ML model and determining the direction of arrival (DoA) of the sound, as well as coordinating communication between all subsystems.

Our design prioritises low latency, computational efficiency and low power consumption to meet our constraints.

2.2.2.1) The ESP32-S3 Microcontroller
The role of the microcontroller is to:

- Receive audio signals via I2S and DMA from the digital microphones.
- Execute FFT (Fast fourier transform) to convert time-domain samples into frequency domain values for classification, execute MFCC (Mel-Frequency Cepstral Coefficients) extraction to generate features typically used for audio classification, do cross correlation (for angle determination) and execute ML inference on the MFCC features for classification.
- Send results of the classification and localisation to the Output Subsystem.
- Coordinate timing and synchronisation between microphones.

We chose the ESP32-S3 microcontroller for a variety of reasons:

- It includes dual core processing, which is required for handling DSP and control tasks in parallel.

- It supports I2S audio interface for digital microphones.
- It has sufficient RAM and flash for DSP buffers and ML models.
- It supports TensorFlow Lite Micro for embedded ML inference, which is a big part of this project because we want our system to work independent of an internet connection.
- It provides GPIO, UART and SPI for communication with other subsystems, and we need GPIO communication for our I/O peripherals.

2.2.2.2) DSP Feature Extraction Pipeline (for Classification):
Raw data is too large and too noisy for it to be used directly for audio classification. DSP is used to extract meaningful features from the signals.

Pipeline stages:

- Audio capture: audio signals/samples are captured via I2S using DMA into circular audio buffers. The circular buffer ensures continuous streaming without loss, making sure that the Processing Unit processes audio in chunks.
- Windowing: Each audio frame is multiplied by a Hamming window function to improve frequency resolution and reduce spectral leakage.
- FFT (Fast Fourier Transform): This step converts time domain samples into frequency domain values because frequency domain is more useful for classification purposes. The equation we use in this step is as follows :

  $$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

  Here, x[n] is the input signal and X[k] is the output, in the frequency domain.
- Mel scale frequency band mapping: FFT magnitudes are mapped onto mel scale frequency bands to reduce feature size. We map to the mel scale using the following equation:

  $$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

- MFCC extraction : We use a Discrete Cosine Transform (DCT) to compress mel coefficients into MFCC coefficients, using the equation:

$$c_k = \sum_{n=1}^{M} \log(E_n) \cos\left[\frac{\pi k(n-0.5)}{M}\right]$$

This is again, to produce MFCC features, to help in audio classification.

2.2.2.3) ML Inference Pipeline:
The ML model is used to map the generated MFCC features to sound classes/types.

The model architecture is as follows:
Input: MFCC features(x)
Output: Probability distribution over sound types (y)

The equation we use is : y = fθ (x)
where, θ: Trained weights

Our target size for the model is <= 20 kB, to keep the computation time to a minimum and we want the model to be deployed using Tensorflow Lite Micro so that our system is independent of an internet connection.
The output of the classification will be a class label (eg - speech, bird sound, gun shot).

2.2.2.4) Localisation pipeline:

- Bandpass filtering: This is for us to remove the noise outside the target frequency range.
- Cross correlation: for 2 microphone signals, x1(t) and x2(t):

  $$R_{12}(\tau) = \sum_{n} x_1(n)x_2(n-\tau)$$

  The peak will then give us the time delay (dt)
- Then, we estimate the TDOA (Time difference of arrival) using the dt, using the formula:
  dt = dd/c
  , where x = speed of sound (343 m/s) and dd = distance between microphones.

- Angle estimation : Then, we estimate the final angle using the equation:

θ = arcsin(c. dt/d)
Here, d is the actual measured distance between the mics.
We calculate this for all sets of microphones (ie - mic 1 and 2, 2 and 3 and 1 and 3) and then, average the result.

Requirements	**Verification**
The latency needs to be <100 ms. This is the time difference between the audio capture and output display.	**We capture the log timestamps at audio capture and output generation and compute the difference, make sure its below 100 ms.**
Classification is more than equal to 85 percent and localisation accuracy is +- 85 percent.	We test the system using a labeled audio dataset and calculate accuracy.

2.2.3 Subsystem 3: Display Subsystem:

The Display Subsystem is responsible for presenting the output of the Processing Subsystem to the user in real time. It receives two pieces of information from the ESP32-S3: the predicted sound class from the ML inference pipeline and the estimated direction of arrival from the TDOA localization algorithm. Its role is to translate these outputs into an intuitive visual representation that allows the user to immediately understand both what sound was detected and where it came from.

Component Selection

We selected an SSD1306 128×64 OLED display as the primary output interface. The SSD1306 communicates over I2C, requiring only two signal lines (SDA and SCL) and allowing it to share the bus with other peripherals if needed, minimizing GPIO usage on the ESP32. It operates on 3.3V, making it directly compatible with our power subsystem. The 128×64 pixel resolution is sufficient to display a text label for the detected sound class alongside a simple directional indicator graphic, such as an arrow or arc pointing toward the estimated angle of arrival. The display draws approximately 10–15 mA during active use, which is within the budget of our power subsystem.

In addition to the OLED, a linear array of five LEDs is included to provide a secondary directional output. Each LED corresponds to an angular sector (far left, left, center, right, far right), and the LED corresponding to the estimated direction of the detected sound will illuminate. This provides an immediate low-resolution directional cue that is readable at a glance without requiring the user to read the OLED screen. The LEDs are driven directly from ESP32 GPIO pins through current-limiting resistors sized to draw approximately 5 mA per LED.

Relationship to Other Subsystems

The Display Subsystem is entirely output-driven. It receives classification labels and direction angle values from the Processing Subsystem over I2C (OLED) and GPIO (LEDs), and has no upstream data flow back to the Processing or Sensor Subsystems. It draws power from the 3.3V rail supplied by the Power Subsystem. Its latency contribution to the end-to-end pipeline must remain small enough that the total system latency from audio capture to display update remains under 100 ms as required by the high-level requirements.

Requirements and Verification

● The OLED display must correctly render the predicted sound class label within 10 ms of receiving the classification result from the ESP32.	Hard-code a classification result string (e.g., "Species A") and a direction value into the ESP32 firmware. Toggle a GPIO pin immediately before and immediately after the I2C display write command. Measure the pulse width on an oscilloscope. Confirm the display write completes in under 10 ms. Visually confirm the correct label

	appears on the OLED screen.
● The LED directional array must illuminate the correct LED corresponding to the estimated direction of arrival.	Fix the TDOA localization output in firmware to each of five discrete angle sectors in turn (far left, left, center, right, far right). For each fixed angle, confirm that only the corresponding LED in the array illuminates and that all others remain off. Repeat for all five sectors
● The end-to-end latency from audio capture to display update must be less than 100 ms.	Insert firmware timestamps at the start of DMA audio capture and immediately after the display write completes. Output both timestamps over UART serial. Play a short known audio stimulus and record the timestamp difference. Confirm the difference is less than 100 ms across 20 consecutive trials.

2.2.4 Subsystem 4: Power Subsystem:

The system will be powered from a 5V USB input, which will be regulated down to 3.3V using a low-dropout regulator capable of supplying sufficient current for the ESP32 and peripherals. Since the ESP32 can draw large transient currents, especially during processing, adequate bulk and decoupling capacitors will be placed near the regulator and microcontroller to maintain voltage stability. The 3.3V rail will power the ESP32, two I2S MEMS microphones, and the LED or OLED display used for sound classification and localization output. To reduce electrical noise that could interfere with audio capture, each microphone will include local decoupling capacitors and optional filtering components on its supply line. The PCB will implement a star-ground layout to separate sensitive microphone grounds from high-current digital return paths. WiFi and Bluetooth will be disabled during normal operation to reduce power consumption and prevent current spikes. Overall, the power subsystem is designed to provide clean, stable power to ensure reliable real-time ML inference and accurate sound detection.

Requirement	**Verification**
● The system shall operate from a 5V USB input source.	● Connect to regulated 5V supply and confirm full functionality.
● The 3.3V rail shall remain within ±5% under full load.	● Measure output voltage while system is processing.
● Total current draw shall not exceed 500 mA.	● Measure current using inline ammeter during peak operation.
● The system shall include overcurrent protection.	● Inspect circuit design and verify presence of fuse or regulator protection.
● WiFi and Bluetooth shall be disabled during normal operation to reduce average current consumption.	● Verify firmware configuration and measure current consumption with wireless modules disabled.

2.3 Tolerance Analysis

The highest-risk component of this design is the real-time audio processing and machine learning inference pipeline on the ESP32. Specifically, extracting MFCC features from dual I2S microphones while simultaneously performing sound localization and neural network classification under memory and timing constraints presents the greatest implementation difficulty. The ESP32 has limited RAM, and improper memory management or inefficient DSP implementation could cause buffer overflows, increased latency, or failed inference. Additionally, maintaining synchronized sampling between the two microphones is critical for accurate localization; timing misalignment beyond a few samples could significantly reduce directional accuracy.The localization accuracy depends on:

1. **Microphone spacing (d)**
    - Nominal spacing: 6 cm
    - PCB placement tolerance: ±0.5 mm
2. **Sampling frequency (fs)**
    - Nominal: 44.1 kHz
    - Crystal tolerance: ±50 ppm
3. **Speed of sound (c)**
    - Nominal: 343 m/s
    - Temperature variation (0–30°C): ±5 m/s

Acceptable tolerances include maintaining end-to-end latency below 100 ms and keeping classification accuracy above 85% on the test dataset. Microphone synchronization error must remain within a small fraction of the sampling period (at 16 kHz, approximately 62.5 μs per sample) to preserve localization reliability.

This risk directly relates to the high-level project requirements of real-time operation, multi-class bird sound identification, and accurate source localization. If timing, memory, or processing constraints are not properly managed, the system may fail to meet its accuracy and latency goals.

3. Cost and Schedule

3.1 Cost Analysis

3.1.1 Parts & Materials The process of picking out parts was made through the My.ECE portal, ECE service shop, and online retailers.

Total cost for all: $39.50.

Description	Unit Price	Quantity	Total cost	Manufacturer
ESP32-S3	$15	1	$15	Espressif
Digital I2S MEMS Microphone	$7.50	2	$15	Adafruit
Decoupling Capacitors	$0.10	5	$0.50	Generic
Resistors	$0.30	10	$3	Generic
OLED Display	6.00	1	6.00	Generic

42.50×2.5×120=12,750 per person. Multiply by 3 people: Around 38000 dollars.

4. Ethics and Safety

Ethical Considerations

This project follows the IEEE/ACM Code of Ethics by prioritizing user safety, data privacy, and responsible system design. Since the system performs on-device audio classification and does not store or transmit recorded audio, privacy risks are minimized. WiFi functionality will be disabled during normal operation to ensure no unintended data transmission occurs. The system will only classify predefined environmental sounds and will not attempt speaker identification or surveillance-based tracking. Testing will involve voluntary participants producing simple sounds in controlled environments, and no personally identifiable information will be collected, so IRB approval is not required. Testing will involve recording publicly available bird sounds and outdoor environmental audio without interacting with or disturbing wildlife. No animals will be handled, manipulated, or exposed to harmful stimuli, so IACUC approval is not required.

Safety considerations

This project presents minimal safety risks because it operates at low voltage (5V USB stepped down to 3.3V) and does not involve high-voltage components or hazardous battery chemistries. Electrical safety will be addressed by using properly rated voltage regulators, current protection (such as a fuse), and adequate decoupling to prevent short circuits or overheating. All soldering and PCB assembly will follow standard lab safety procedures, including eye protection, proper ventilation, and careful handling of hot tools. The device contains no mechanical moving parts, sharp edges, or high-temperature elements that could pose a risk to users. Since the system is powered through USB and does not use volatile chemicals or large batteries, risks to end users are minimal. Overall, the project falls under low-risk laboratory electronics work and does not require a separate lab safety manual or specialized battery safety training.

Schedule

Week	Task	Person
Feb 27 - March 7	1. Finalize PCB Design 2. Order Parts 3. Begin Board Assembly 4. Set up ESP32 development environment 5. Get basic I2S audio capture from MEMS microphone on breadboard	1. All 2. Ahaan 3. Ahaan 4. Kavin 5. Om
March 10 - March 14	1. Validate breadboard FFT output over serial 2. Begin MFCC extraction in Python for model training 3. Pass PCB Audit	1. Ahaan 2. Om 3. Kavin

March 17 - March 23	1. Receive and inspect fabricated PCB 2. Begin soldering and assembly 3. Verify 3.3V power rail with multimeter	1. Ahaan 2. Kavin 3. Om
March 24 - March 28	1. Complete PCB bring-up 2. Confirm I2S DMA audio capture from both microphones on PCB 3. Begin porting DSP pipeline to ESP32 firmware	1. Kavin 2. Ahaan 3. Om
March 31 - April 4	1. Deploy TFLite Micro model on ESP32 2. Demonstrate end-to-end classification of at least one bird species 3. Measure inference latency	1. Om 2. Kavin 3. Ahaan
April 7 - April 11	1. Integrate TDOA localization algorithm 2. Validate ±5° directional accuracy against known source positions 3. Connect LED/OLED display output	1. Om 2. Ahaan 3. Kavin
April 14 - April 18	1. Run full accuracy	1. Om

	test on held-out bird vocalization dataset 2. Confirm ≥85% classification accuracy and <100ms latency 3. Complete mock demo and presentation	2. Kavin 3. Ahaan
April 21 - April 25	1. Final Demo 2. Final system validation 3. Prepare demo setup 4. present complete working system	1. All 2. All 3. All 4. All

References

[1] Espressif Systems, *ESP32-WROOM-32 Datasheet*, Espressif Systems, 2023.

[2] Espressif Systems, *ESP32 Technical Reference Manual*, Espressif Systems, 2023.

[3] STMicroelectronics (or manufacturer of your MEMS mic), *I2S Digital MEMS Microphone Datasheet*, 2023. *(Replace with your actual microphone part number and manufacturer.)*

[4] TensorFlow, “TensorFlow Lite for Microcontrollers,” TensorFlow.org. [Online]. Available: https://www.tensorflow.org/lite/microcontrollers

[5] IEEE, “IEEE Code of Ethics,” IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[6] Idea for MFCC feature extraction: Speaker Recognition for Mobile User Authentication: An Android Solution , K. Brunet, K. Taam , September 2013 https://www.researchgate.net/publication/257365356_Speaker_Recognition_for_Mobile_User_Authentication_An_Android_Solution