

# ESP32 Microphone – Proof of Concept Audio Streaming Device

---

## Abstract

The ESP32 Microphone project is a proof-of-concept embedded system that demonstrates real-time digital audio capture and streaming using an ESP32-WROOM-32 microcontroller. Audio is sampled through a MEMS microphone (INMP441) over the I<sup>2</sup>S interface and transmitted via UDP to a receiver on the same local network. The device includes Wi-Fi provisioning through an HTTP captive portal, non-volatile credential storage, and UART-based debugging functionality.

This project was developed as a personal research and learning initiative in embedded systems programming using the ESP-IDF framework in C.

## 1. Project Purpose

The main objective of the ESP32 Microphone project was to explore embedded systems development with an emphasis on communication protocols, concurrency, and low-level hardware integration. The system acts as a network-connected microphone capable of streaming raw audio data to a UDP receiver (e.g., using ffplay). Through experimentation with ESP-IDF's APIs and FreeRTOS multitasking, the project integrates audio capture, Wi-Fi networking, UART communication, and non-volatile storage in a single firmware environment.

While not designed as a commercial product, the implementation successfully demonstrates a functioning prototype of a simple “network spy microphone” — a device that can automatically connect to Wi-Fi, capture audio, and stream it to a server on the same LAN.

## 2. Technologies and Development Environment

Programming Language: C

Framework: ESP-IDF v5.0

Microcontroller: ESP32-WROOM-32

Microphone Module: INMP441 (digital MEMS microphone, I<sup>2</sup>S interface)

Operating System: FreeRTOS (bundled with ESP-IDF)

Protocols Used: I<sup>2</sup>S, UDP, HTTP, UART, NVS (Non-Volatile Storage), WLAN

Host Environment: Windows 11 with Visual Studio Code

Build System: idf.py (ESP-IDF build and flash utility)

The project uses the standard ESP-IDF template structure, with all libraries and user modules organized under main/.

### 3. Hardware Setup

The system consists of an ESP32-WROOM-32 microcontroller connected to an INMP441 MEMS microphone, an LED status indicator, and a reset pin for memory control. Power can be supplied via USB or a Li-ion battery through a TP4056 charging module.

INMP441 Pin	ESP32 Pin	Function
L/R	GND (breadboard)	Channel selection
WS	GPIO33	Word Select (LRCLK)
SCK	GPIO14	Bit Clock (BCLK)
SD	GPIO25	Serial Data input
VDD	3V3	Power supply
GND	GND (breadboard)	Ground

Additional connections:

- GPIO18 → short to GND resets NVS (clears stored Wi-Fi credentials)
- GPIO13 → LED indicator (ON when connected to Wi-Fi)
- TP4056 → battery charging and protection circuit for mobile operation

### 4. Software Architecture

The firmware follows a modular structure for maintainability and clarity. Each module encapsulates a subsystem such as Wi-Fi, UDP, I<sup>2</sup>S, or UART. The project leverages FreeRTOS tasks for concurrent execution.

### 5. System Logic and Operation

Upon startup, the ESP32 executes the following sequence:

1. Initialization: system setup, GPIO configuration, and peripheral initialization.
2. Wi-Fi Provisioning: AP mode, HTTP captive portal, credential storage.
3. Connection Phase: STA mode activation and LED indication.
4. Audio Sampling and Streaming: continuous data capture and UDP transmission.
5. Debugging and Control: UART interface for packet size and diagnostics.
6. Reset Behavior: GPIO18 to GND clears credentials and restarts provisioning.

## **6. Debugging and Configuration**

The UART interface serves as a flexible debugging tool. Through it, developers can monitor system logs in real-time, adjust the UDP packet size, reset tasks, or test runtime stability.

## **7. Future Improvements**

Planned and potential improvements include MQTT integration for remote streaming, addition of a speaker module for bidirectional audio and encrypted communication

## **8. Author**

Developed by: Igor Adamowicz

Affiliation: Wrocław University of Science and Technology

Field of Study: Cybersecurity

Date: 2025