Secure Communication Terminal

Introduction

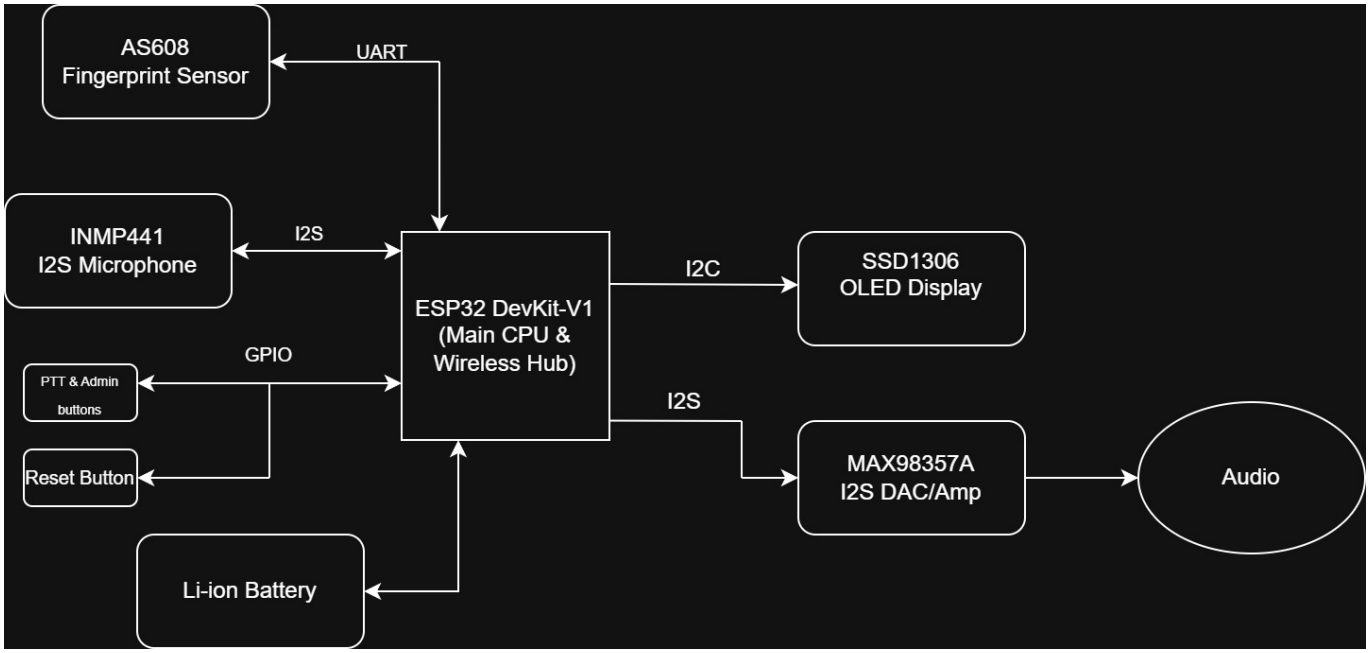
The **Secure Communication Terminal** project is a digital, secured wireless communication device. Unlike a classic radio station, the device allows real-time voice capture, P2P transmission, and playback only if the user passes a **biometric authentication** filter. The system integrates a permission-based access hierarchy (Admin vs. User) and is controlled by an ESP32 microcontroller.

What is its purpose: The main goal is to build a robust physical product, a sort of industrial prototype, capable of combining modern access control technologies (fingerprint sensor) with high-fidelity digital audio processing. At a technical level, the project demonstrates mastery of microcontroller architectures through the simultaneous integration of several studied complex protocols (UART, I2C, I2S) and the efficient use of hardware interrupts for the system's state machine.

What was the starting idea: The idea stemmed from the main vulnerability of conventional analog radio stations: lack of security. Anyone owning a station on the same frequency can listen or transmit. I thought about how I could implement a fundamental principle of cybersecurity (*Access Control* based on “Something you are”) directly into the physical environment, transforming a simple communication station into a strictly restricted data terminal.

General Design

To illustrate the architecture of the **Secure Communication Terminal**, I have created a block diagram highlighting the hardware components and the data flow (communication protocols) between the peripherals and the central processing unit.



User Roles & Access Control

The terminal supports two types of users: **Admin** and **General User**.

- The Admin fingerprint is permanently stored and cannot be modified or overwritten.
- Only the Admin has the authority to enroll or remove a General User's fingerprint.
- **Add User:** The Admin authenticates by scanning their fingerprint, presses the action button briefly, and then the new user introduces their fingerprint.
- **Remove User:** The Admin authenticates and long-presses the action button for 5 seconds to wipe the General User from the database.

Usage & Communication Flow

The OLED screen acts as the main interface, displaying the current communication status. During an incoming wireless transmission, an '**Incoming Transmission**' message is shown on the screen.

If the device is unlocked (a user is logged in), the incoming audio is actively played through the speaker. Once authenticated, the user can receive and transmit freely for a **2-minute session** before the system automatically times out, locks itself, and requires re-authentication.

The reset button must be held for 5 seconds to perform a full reset of the system.

Hardware Design

Component	Quantity	Description	Interface

—	—	—	—
ESP32 DevKit V1	2	Main Microcontroller (WROOM-32)	-
AS608 Sensor	2	Optical Biometric Fingerprint Sensor	UART
INMP441 Mic	2	Digital Omnidirectional Microphone	I2S0
MAX98357A Amp	2	I2S DAC + Class D Audio Amplifier	I2S1
SSD1306 OLED	2	0.96” Monochrome Display (128×64)	I2C
Mini Speaker	2	20x30mm Rectangular Speaker (1W)	Analog
Tactile Buttons	6	PTT, Admin Mode, Reset	GPIO
Battery Holder	2	4 x AA Slots Enclosure	Power
NiMH AA Batteries	8	1.2V Rechargeable Cells (4.8V pack per terminal)	Power
Custom Enclosure	2	3D Printed Case & Breadboard Assembly	Mechanical
DuPont Wires	Set	Male-to-Male / Male-to-Female Jumpers	Wiring

System Architecture & Power Management: The hardware architecture centers on the ESP32, managing biometric security via UART and real-time audio through concurrent I2S buses. A critical design decision was made to ensure portability and prevent brownout resets during high-current audio playback.

The system is powered by a 4-cell AA battery holder utilizing NiMH rechargeable batteries (providing a stable ~4.8V). This power source is connected directly to the ESP32's `VIN` pin and the MAX98357A amplifier. This maximizes the audio output volume (staying safely below the amplifier's 5.5V absolute maximum rating) while allowing the ESP32's internal 3.3V regulator to safely power the logic-level peripherals (OLED, Mic, Fingerprint Sensor) without being overloaded by audio transients.

Detailed Pin Mapping & Motivation:

Component	Peripheral Pin	ESP32 Pin	Signal Type	Design Motivation
Power Supply	Plus (+) Bat.	VIN	Power (4.8V)	System power. Feeds Amp directly and ESP32 regulator.
	Minus (-) Bat.	GND	Ground	Common reference ground.
Fingerprint (AS608)	Pin 3 (Green)	3V3	Power (3.3V)	Safe operating voltage for the sensor.
	Pin 4 (Yellow)	D16 (RX2)	UART TX	Dedicated hardware UART2 for reliable biometric data.
	Pin 5 (Black)	D17 (TX2)	UART RX	Dedicated hardware UART2.
Microphone (INMP441)	VDD	3V3	Power (3.3V)	Native digital power.
	L/R	GND	Config	Tied to GND to configure transmission on the Left Channel (Mono).
	WS	D33	I2S Clock	Allocated to standard output-capable pins for I2S0 Master mode.
	SCK	D18	I2S BClock	
	SD	D32	I2S Data	
Amplifier (MAX98357A)	VIN	VIN	Power (4.8V)	Powered directly from batteries to prevent ESP32 brownouts.
	LRC / WS	D26	I2S Clock	Allocated to the secondary I2S1 bus for independent audio output streaming.
	BCLK	D27	I2S BClock	
	DIN	D14	I2S Data	
Speaker	Positive (+)	Amp OUT+	Analog	Driven directly by the Class D Amplifier for high-efficiency output.
	Negative (-)	Amp OUT-	Analog	

OLED (SSD1306)	VCC	3V3	Power (3.3V)	Standard logic power.
	SDA	D21	I2C Data	Native hardware I2C pins for maximum compatibility with the Wire library.
	SCL	D22	I2C Clock	

Software Design

The project's software is built around a robust **Finite State Machine (FSM)** architecture, utilizing **FreeRTOS** capabilities for multitasking and asynchronous interrupt processing. This allows the separation of time-critical tasks (audio streaming) from low-priority tasks (UI updates).

Libraries Used

* **esp_now.h:** Chosen over classic Wi-Fi or Bluetooth to eliminate router dependency and ensure ultra-low latency, which is essential for voice transmissions (Voice-over-Radio). * **driver/i2s_std.h (ESP32 IDF):** Used for direct control of the I2S bus. Unlike a standard analog ADC/DAC, I2S allows pure digital reading of the microphone and digital control of the amplifier, ensuring vastly superior audio quality free of circuit interference. * **Adafruit_Fingerprint.h:** Chosen for the efficient abstraction of UART communication with the AS608 sensor, greatly simplifying the complex mathematical operations involved in storing and matching biometric matrices. * **freertos/queue.h:** Vital for decoupling. It allows the separation of the radio reception process (ISR) from the audio playback process, using a memory queue to guarantee continuous voice playback without blocking.

Finite State Machine (FSM) and Logic Flow

The application backbone runs through transitions between 4 distinct states:

* **STATE_LOCKED / ST_LOCKED:** Default state. The system is completely isolated. The I2S output is muted, and the system polls the AS608 sensor via UART. * **STATE_UNLOCKED / ST_UNLOCKED:** Reached after a valid fingerprint match. A timer is started (2 minutes). * **STATE_WAIT_ADMIN / ST_ADMIN:** An intermediate state triggered when the Admin ID (ID 1) is recognized. Enables 'Enroll' and 'Delete' functions via the Action Button. Features an automatic 5-minute timeout. * **STATE_LIVE_STREAM / ST_TRANSMIT:** The “Walkie-Talkie” communication mode. Triggered by the PTT button (GPIO Interrupt), the station switches between Receiver and Transmitter modes (Half-Duplex). Audio is sampled via I2S and sent through ESP-NOW.

To avoid needing excessive physical buttons, **temporal multiplexing** is used for the Action Button:

- **Short Press (< 2s):** If in `ST_ADMIN`, it triggers the `fingerprintEnroll()` function to add the General User via a 3-step software routine.
- **Long Press (> 5s):** If in `ST_ADMIN`, it triggers `fingerprintDelete(USER_ID)` to wipe the database.

Hardware-Level Encryption (AES-128)

To prevent unauthorized interception of the radio traffic (packet sniffing), the system implements ESP-NOW's native **AES-128** encryption at the MAC layer.

- **Symmetric Keying:** A 16-byte secret key (`secretKey`) is hardcoded and shared between the ALPHA and BRAVO terminals. This acts as both the Primary Master Key (PMK) and the Local Master Key (LMK).
- **Secure Payload:** By setting `peerInfo.encrypt = true` during the peer registration phase, the ESP32's Wi-Fi hardware automatically encrypts the outgoing 240-byte audio payloads and decrypts them upon arrival. This zero-overhead hardware encryption ensures that the P2P voice stream remains strictly confidential.

Communication Protocol & Audio Flow

A major technical challenge was the limitation of the **ESP-NOW** hardware protocol, which strictly supports **250 bytes per payload**. System calibration was achieved by reading the microphone stream (which is natively 32-bit) and truncating/packing it into a buffer of exactly **120 samples of 16-bit**. The result is a packet of exactly **240 bytes**, maximizing the available audio bandwidth without hitting the ceiling that would cause packet loss.

Received packets are captured through an interrupt routine (`OnDataRecv`) triggered asynchronously by the hardware, using the `xQueueSendFromISR` command to safely place the data into the playback buffer.

Critical System Optimizations

* **Where:** On the I2C bus (OLED Display). * **How:** The `display.display()` command was strictly restricted only to the moments when the device changes its state (e.g., transition from TX to RX). * **Why:** I2C is a much too slow bus compared to the frequency of the incoming radio packets (dozens per second). Updating the screen for every packet would have led to “CPU starvation”, severely fragmenting the audio playback fluency.

* **Where:** Background Noise Management (I2S Speaker). * **How:** In the absence of a valid radio signal in the FreeRTOS queue, the system constantly injects an array of zeros into the amplifier. * **Why:** This hardware method completely eliminates electrostatic hiss and white noise while in standby, maintaining absolute “digital silence” until the next transmission.

Results

Demo Link : https://youtube.com/shorts/ny9w-C_flPQ?is=Uj0ZvqpBS-vCuD1a

Download

Project files can be found here:
https://github.com/Catalin951/Secure-Communication-Terminal/tree/main