Internal use only

Infonique Voice Recognition

Prepared by	Date	Version
Bing Ran	03/20/2025	1.0

1 Abstract

This document details the implementation of a voice recognition system using the ESP32, covering both hardware and software aspects. The system is designed to recognize voice commands, process them, and respond with spoken feedback using pre-recorded audio files. The hardware includes an ESP32 Dev module, a VC-02 voice recognition module, an I2S audio output system, and an OLED display for visual feedback. The software integrates voice detection, serial communication, and audio playback . This document provides an in-depth exploration of system design, firmware development, and practical considerations, offering a comprehensive guide for developers interested in embedded voice recognition applications.

Document History

Date	Rev	Modifier	Changes
20-March-2025	1.0	Bing Ran	First Draft

Contents

Table of Figures

Table of table

1 Introduction

1.1 Overview of Sound Recognition Technology

Voice recognition technology allows electronic devices to interpret and respond to audio signals, enabling voice commands, environmental sound detection, and real-time interaction. This technology is widely used in applications such as voice assistants (e.g., Alexa, Google Assistant), security systems, and accessibility tools for individuals with disabilities.

At its core, voice recognition involves capturing audio signals, processing them to extract features, and matching them to predefined patterns. With the advancement of machine learning and edge computing, microcontrollers like the ESP32 can now perform sound recognition efficiently without requiring cloud-based processing.

1.2 Importance of Embedded Sound Recognition

Embedded voice recognition plays a crucial role in creating smart and interactive systems. Unlike cloud-based solutions, embedded sound recognition processes audio locally on the device, offering advantages such as:

- Faster response time (no internet dependency)
- Better privacy and security (audio data is not sent to external servers)
- Lower power consumption (suitable for battery-powered devices)

For students, learning embedded voice recognition provides valuable insights into digital signal processing (DSP), machine learning on microcontrollers, and real-world embedded system development.

1.3 Objectives of This Project

The goal of this project is to provide students with hands-on experience in implementing a voice recognition system using the ESP32 microcontroller. This system will:

1. Recognize voice commands using the VC-02 voice module
2. Retrieve and announce temperature & humidity data from an SHT31 sensor
3. Provide audio responses via I2S speaker output
4. Display real-time readings on an SSD1306 OLED screen
5. Use LittleFS for storing and playing pre-recorded audio files

By completing this project, students will learn about microcontroller programming, interfacing with hardware components, and implementing embedded voice recognition in practical applications.

2. Hardware

The selection of hardware components plays a crucial role in ensuring smooth operation and performance of the embedded voice recognition system. Each component has been chosen to provide efficient processing, reliable voice recognition, clear audio output, real-time display feedback, and accurate environmental sensing.

2.1 ESP32 – Microcontroller for Processing

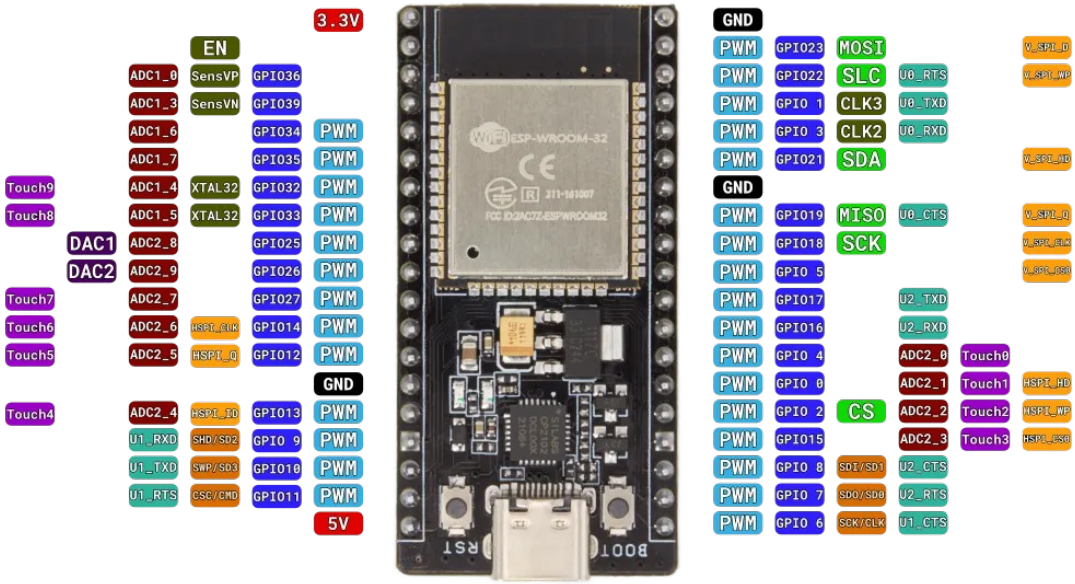


Figure 1: ESP32 Dev Module

The ESP32 is the core processing unit of this project, responsible for handling all data processing, communication, and control operations. It is chosen due to its:

- Dual-core processor for handling multiple tasks efficiently
- Wi-Fi & Bluetooth capabilities (though not used in this project, they allow future expansion)
- Integrated I2S, I2C, and UART interfaces for seamless connection with peripherals
- Low-power operation suitable for battery-powered applications

The ESP32 manages the voice recognition module (VC-02), processes temperature and humidity sensor readings (SHT31), controls the OLED display (SSD1306), and plays audio responses using the MAX98357A I2S audio amplifier.

2.2 VC-02 Voice Recognition Module – Recognizing Voice Commands

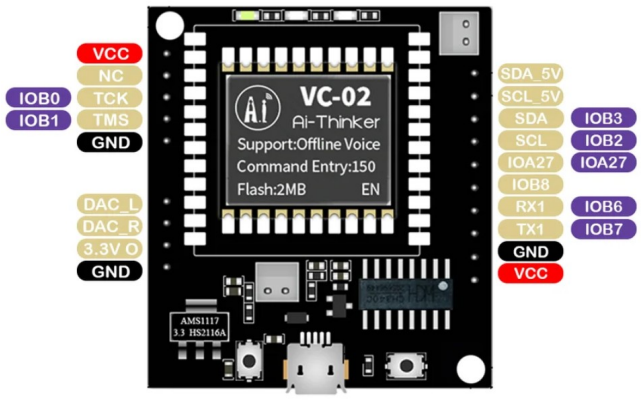


Figure 2: VC-02 Module

The VC-02 module is a compact, pre-trained voice recognition module capable of recognizing custom voice commands. It is selected due to:

- Built-in offline voice recognition (no need for internet access)
- Customizable wake words & commands
- UART interface for easy communication with ESP32
- Low power consumption

Function in the Project

- Detects specific voice commands like `"Hi Pudding"`, `"Humidity"`, “Temperature” and `"Siren"`
- Sends recognized command data to ESP32 via UART

2.3 I2S Audio Output (MAX98357A + Speaker) – Playing Responses

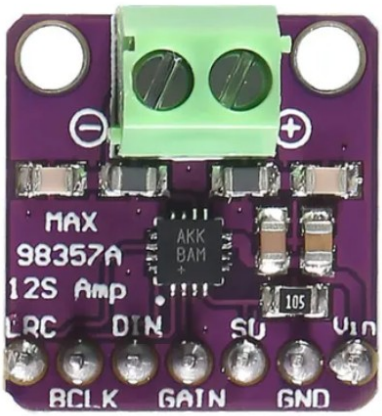


Figure 2: MAX98357A

The MAX98357A is a digital-to-analog converter (DAC) and audio amplifier that converts digital I2S signals into analog audio output, which is played through a connected speaker.

Reasons for Choosing MAX98357A

- I2S support for high-quality audio playback
- Built-in amplifier eliminates the need for additional circuitry
- Compact and low-power design

Function in the Project

- Receives pre-recorded WAV audio files stored in ESP32's LittleFS storage
- Converts digital audio signals into analog signals
- Outputs sound through a connected speaker

2.4 SSD1306 OLED Display – Visual Feedback

Figure 4: SSD1306 OLED Display

The SSD1306 OLED display is a 128x64 pixel monochrome display used to show real-time sensor readings and system status.

Reasons for Choosing SSD1306

- Low power consumption (suitable for embedded applications)
- I2C communication (only requires 2 pins, SDA & SCL)
- Clear, high-contrast display for easy readability

Function in the Project

- Displays real-time temperature and humidity readings
- Provides visual confirmation of recognized voice commands

2.5 SHT31 Temperature & Humidity Sensor – Sensor Data Acquisition

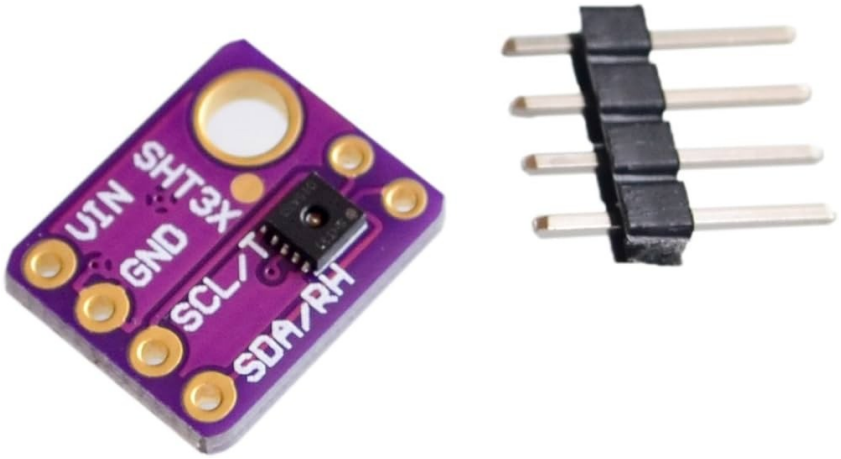


Figure 5: SHT31

The SHT31 is a highly accurate temperature and humidity sensor that communicates with the ESP32 over I2C.

Reasons for Choosing SHT31

- High accuracy (±0.3°C for temperature, ±2% for humidity)
- Fast response time
- I2C communication (easy interface with ESP32)

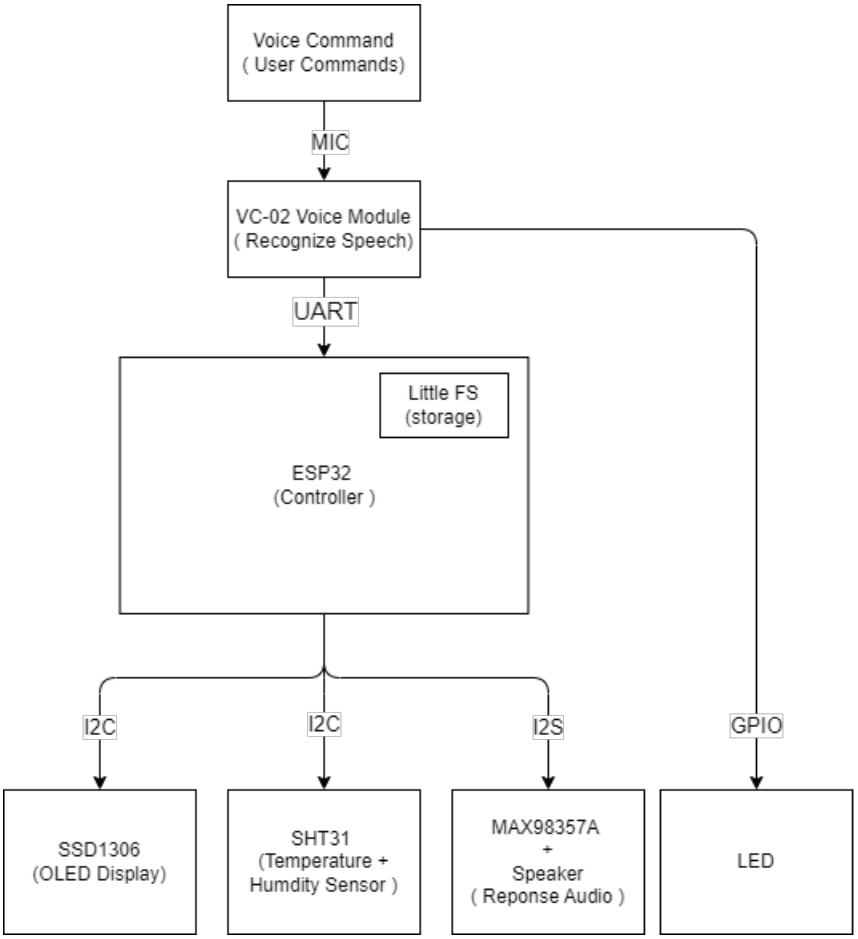
Function in the Project

- Measures temperature and humidity in real-time
- Sends data to the ESP32 for processing and display
- Enables the audio response system to announce current weather conditions

3 Setting up Development Environment

This guide will help students set up their iSEB voice recognition. We will use the VC-02 sound recognition module to recognize voice comand and play response using I2S speaker.

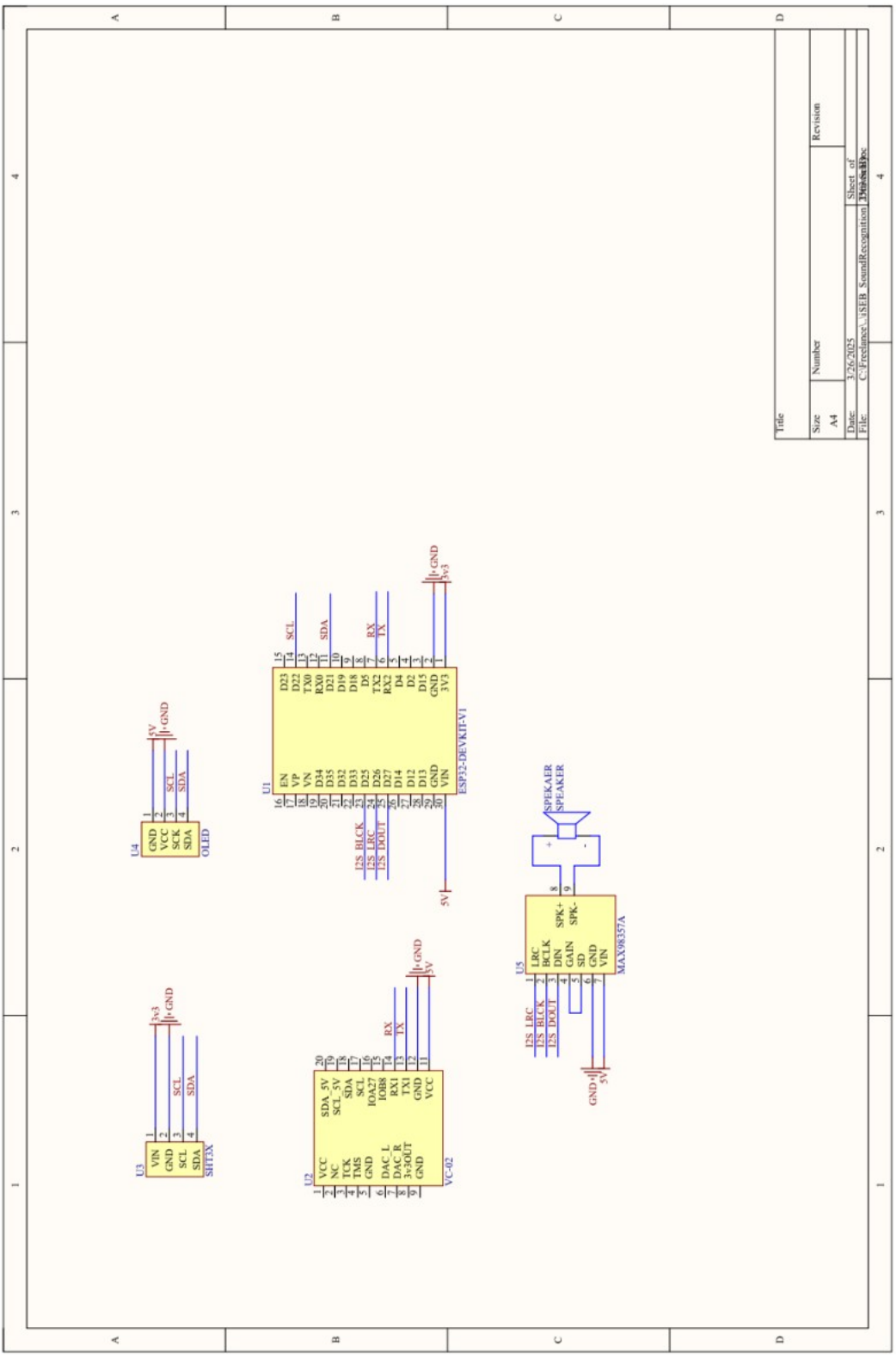
3.1 Block Diagram



The system consists of the following primary components:

1. **ESP32 (Main Controller)** – Handles communication between all peripherals.
2. **VC-02 Voice Recognition Module** – Recognizes voice commands and sends them to the ESP32 via UART.
3. **SHT31 Temperature & Humidity Sensor** – Measures real-time environmental data and sends it to ESP32 via I2C.
4. **SSD1306 OLED Display** – Provides visual feedback on system status and sensor readings.
5. **MAX98357A I2S Audio Output + Speaker** – Plays voice responses based on recognized commands.
6. **LittleFS (ESP32 Internal Storage)** – Stores pre-recorded **WAV** files used for audio playback.

3.2 Schematic of the Infonique voice recognition



3.3 Pinout connection

Component	ESP32 Pin	Function
VC-02 TX	GPIO16	UART RX (Receive voice commands)
VC-02 RX	GPIO17	UART TX (Send data if required)
I2S BCLK (MAX98357A)	GPIO25	I2S Bit Clock
I2S LRC (MAX98357A)	GPIO26	I2S Left-Right Clock
I2S DOUT (MAX98357A)	GPIO27	I2S Data Output
SSD1306 OLED SDA	GPIO21	I2C Data
SSD1306 OLED SCL	GPIO22	I2C Clock
SHT31 SDA	GPIO21	I2C Data (Shared with OLED)
SHT31 SCL	GPIO22	I2C Clock (Shared with OLED)

Table 1: ESP32 Pin Connection

3.4 Hardware compoenents

Component	Description
ESP32 Board	The main microcontroller to process voice commands
VC-02 Voice Recognition Module	Recognizes voice commands and sends them to ESP32
VC-02 Programmer	Used to upload firmware and custom voice commands to the VC-02 module.
MAX98357A I2S Amplifier	Plays sound responses
Speaker (8Ω, 3W)	Outputs the voice responses
SSD1306 OLED Display	Displays text output
SHT31 Sensor	Measures temperature & humidity
Jumper Wires	Connects components together
Micro USB Cable	Uploads code to ESP32
Breadboard	Allows easy prototyping by connecting components without soldering.

Table 2: Hardware Component

3.3 Development Environment

The guide will help student to set up their infonique sound recognitoin development environment .
Software Setup

Step 1 : Install Arduino IDE to program the ESP32.

- Download Arduino IDE from: https://www.arduino.cc/en/software
- Install it on your computer.
- This prototype is built with Aduino IDE 2.3.2

Step 2: Install ESP32 Board in Arduino IDE

1. Open Arduino IDE.
2. Go to File → Preferences.
3. Find "Additional Board Manager URLs" and paste:https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
4. Click OK.
5. Go to Tools → Board → Boards Manager.
6. In the search bar, type ESP32.
7. Click Install for "ESP32 by Espressif Systems".
8. This prototype is using version 3.1.3.

Step 3 : Install Required Libaries

In Arduino IDE, go to:
Sketch → Include Library → Manage Libraries, then install:

1. Adafruit_SSD1306.h(tested with v2.5.10) → For the OLED display
2. Adafruit_GFX.h (tested with V1.12.0) → Graphics library for OLED
3. Adafruit_SHT31.h (tested with v2.2.2)→ For temperature & humidity sensor
4. AudioFileSourceLittleFS.h (tested with ESP8266Audio v2.0.0) → To read audio files
5. AudioGeneratorWAV.h (tested with ESP8266Audio v2.0.0)→ To play WAV files
6. AudioOutputI2S.h (tested with ESP8266Audio v2.0.0) → To output sound
7. LittleFS.h,Wire.h and HardwareSerial.h is standard ESP32 library.

Step4: Configure LittleFS for WAV file storage.

LittleFS stores the WAV files on ESP32's flash memory.

1. Enable LittleFS in Arduino IDE:
    - Go to Tools → Partition Scheme → "Huge APP (NO OTA 2MB APP/2MB SPIFFS)".
    - This allocates more space for LittleFS.
2. Upload WAV Files to ESP32:
    - Place WAV files (`hi.wav`, `temp.wav`, etc.) in `data/` inside your project folder.
    - Generetae the littlefs.img by calling the cmd
        - ""ADD UR PATH'\iSEB_SoundRecognition\Software\Tools\mklittlefs\mklittlefs.exe" -c data/ -p 256 -b 4096 -s 0x1E0000 littlefs.img
        - eg : "C:\iSEB_SoundRecognition\Software\Tools\mklittlefs\mklittlefs.exe" -c data/ -p 256 -b 4096 -s 0x1E0000 littlefs.img
    - Upload the littlefs.img by calling.

- python -m esptool --chip esp32 --port “YOUR COMP PORT “ write_flash 0x210000 littlefs.img
- eg: python -m esptool --chip esp32 --port COM6 write_flash 0x210000 littlefs.img

Step 5 : Configure and upload binary to VC-02 voice recognition module

The details can refer to the link below:
https://docs.ai-thinker.com/en/voice_module
The **VC-02** voice recognition module requires a firmware update or custom binary upload to work correctly. We can use the VC-02 usb programmer to program it with following connection:

- VC-02 Module → VC-02 Programmer
- VC-02 Programmer → PC via USB

The programmer has built-in connections, so no extra wiring is needed. We can just upload the binary file with the following steps.

- Open VC_EVB.exe
- Select COM Port (Check Device Manager for your USB adapter)
- Click "Open COM"
- Click "Load Firmware" and select the .bin file
- Click "Start Upgrade"
- Wait for the process to finish (takes ~1 min)
- Click "Reset Module" or manually press RESET on VC-02

Step 6: Connecting the hardware

Use jumper wires to connect the components.

- Connect VC-02 (voice module)

VC-02 Pin	ESP32
TX (Output)	GPIO 16 (RX)
RX (Input)	GPIO 17 (TX)
GND	GND
VCC	3.3V

- Connect MAX98537A (Speaker Amplifier)

MAX98357A Pin	ESP32
BCLK (Bit Clock)	GPIO 25
LRC (Word Select)	GPIO 26
DIN (Data Input)	GPIO 27
GND	GND
VIN	5V

- Connect SSD1306 OLED Display

OLED Pin	Connect to ESP32
SDA	GPIO 21
SCL	GPIO 22
GND	GND
VCC	3.3V

- Connect SHT31 (Temperature & Humidity Sensor)

SHT31 Pin	*Connect to ESP32*
SDA	*GPIO 21*
SCL	*GPIO 22*
GND	*GND*
VCC	*3.3V*

Step 7 Run your firt voice command.

- The Upload code to esp32
- ESP32 will retrieve the reading from SHT31 and display at SSD1306 OLED display.
- VC-02 will detect the pretrained voice command and send uart communication to ESP32.
- ESP32 will decode the command and retreive the wav file from litttle FS and reply through MAX98537A