Informe Técnico: Sistema de Reconocimiento de Voz ESP32

Solución Implementada: Reconocimiento de Voz Simple sin Dependencias Externas

Este documento presenta la solución implementada para el sistema de reconocimiento de voz en ESP32 con micrófono INMP441 y control de LEDs. La solución utiliza un enfoque simplificado sin dependencias de TensorFlow ni otras bibliotecas externas.

Resumen Ejecutivo

La solución desarrollada cumple con todos los requisitos funcionales solicitados por el cliente:

- Reconocimiento de 4 comandos o fonemas
- Activación de LEDs correspondientes a cada comando
- Compatible con micrófono INMP441 (I2S)
- Implementación en ESP32 con PlatformIO

El sistema utiliza análisis de frecuencia simplificado basado en:

1. Análisis temporal de la señal de audio
2. Detección de cruces por cero para estimación de frecuencia
3. Distribución de energía en bandas de frecuencia
4. Comparación de patrones con perfiles pre-calibrados

Características Técnicas

Especificaciones del Sistema

- **Microcontrolador**: ESP32
- **Interfaz de Audio**: I2S (INMP441)
- **Tasa de Muestreo**: 16 kHz
- **Resolución**: 16 bits
- **Bandas de Frecuencia**: 4 (baja, media, alta, muy alta)
- **Tiempo de Respuesta**: ~1 segundo
- **Consumo de Memoria**: Mínimo (sin bibliotecas externas)

Ventajas de la Implementación

1. **Reducción de complejidad**: Sistema más fácil de comprender y mantener
2. **Calibración integrada**: El sistema incluye un modo de calibración para personalización

Funcionamiento del Sistema

Principio de Operación

La solución implementa un algoritmo basado en análisis de frecuencia simplificado:

1. **Captura de audio**: El micrófono INMP441 captura audio a 16 kHz vía I2S
2. **Detección de actividad de voz**: Se calcula la energía total para identificar cuando hay voz
3. **Extracción de características**:
    - Se divide la señal en 4 segmentos temporales
    - Para cada segmento se calcula:
        - Energía relativa
        - Cruces por cero (estimación de frecuencia dominante)
4. **Reconocimiento de patrones**:
    - Comparación de características con perfiles preestablecidos para cada palabra
    - Cálculo de similitud usando distancia relativa
5. **Toma de decisión**:
    - Selección del comando con mayor similitud superior al umbral
    - Activación del LED correspondiente

Arquitectura del Sistema

Componentes Hardware

- **ESP32**: Procesamiento principal
- **INMP441**: Micrófono digital I2S
- **LEDs**: 4 LEDs para indicación visual (con resistencias)

Componentes Software

- **Configuración I2S**: Inicialización del bus I2S para captura de audio
- **Procesamiento de Audio**: Análisis simplificado sin FFT
- **Detección de Comandos**: Comparación con perfiles predefinidos
- **Modo de Calibración**: Sistema para personalizar perfiles de voz
- **Control de LEDs**: Interfaz de salida para indicación visual

Modos de Operación

1. **Modo Normal**: Reconocimiento continuo de comandos
2. **Modo Calibración**: Entrada por GPIO15 para calibrar perfiles de voz

Instructivo de Implementación para PlatformIO

Requisitos Previos

- Visual Studio Code con extensión PlatformIO instalada
- Placa ESP32 (WROOM-32 o similar)
- Micrófono INMP441
- 4 LEDs con resistencias limitadoras (220-330Ω)
- Cables para conexiones

Configuración de PlatformIO

1. Crear un nuevo proyecto en PlatformIO
2. Seleccionar placa ESP32 Dev Module
3. Usar el framework Arduino
4. Configurar platformio.ini según se indica más adelante en este documento

Estructura del Proyecto

```
esp32_voice_recognition/
├── platformio.ini
├── src/
│   └── main.cpp          # Código principal del proyecto
├── include/          # Carpeta para encabezados (vacía en este proyecto)
└── lib/              # Bibliotecas adicionales (no necesarias en este proyecto)
```

Pasos de Implementación

1. **Crear un Nuevo Proyecto**:
    - Abrir PlatformIO Home en VS Code
    - Seleccionar "New Project"
    - Nombre: esp32_voice_recognition
    - Board: Espressif ESP32 Dev Module
    - Framework: Arduino
2. **Configurar el Archivo platformio.ini**:
    - Reemplazar contenido con la configuración proporcionada
3. **Código Fuente**:
    - Reemplazar el contenido de src/main.cpp con el código proporcionado
4. **Compilar y Cargar**:
    - Conectar el ESP32 al ordenador
    - Presionar el botón "Upload" en PlatformIO
    - Esperar a que termine la compilación y carga
5. **Verificar Funcionamiento**:
    - Abrir el Monitor Serial (115200 bauds)
    - Verificar mensajes de inicialización correcta
    - Probar comandos de voz

Calibración del Sistema

Para personalizar el sistema a su voz o a palabras específicas:

1. **Entrar en Modo Calibración**:
    - Conectar GPIO15 a GND
    - Reiniciar el ESP32 (botón Reset o desconectar y reconectar)
2. **Proceso de Calibración**:
    - Seguir instrucciones en el monitor serial
    - Pronunciar cada palabra cuando se indique
    - Anotar los valores generados para cada palabra
3. **Aplicar Calibración Personalizada**:
    - Modificar los valores en la estructura VoiceCommand commands[4] en el código
    - Recompilar y cargar el nuevo firmware
4. **Ajustar Umbrales (si es necesario)**:
    - ENERGY_THRESHOLD: Umbral de detección de voz, ajustar según entorno
    - bestScore en función recognizeCommand(): Umbral de precisión (0.65 por defecto)

Mantenimiento y Solución de Problemas

Problemas Comunes

1. **No se detecta audio**:
    - Verificar conexiones del micrófono INMP441
    - Comprobar que L/R está conectado a GND
    - Reducir ENERGY_THRESHOLD si el entorno es silencioso
2. **Falsos positivos**:
    - Aumentar valor bestScore en función recognizeCommand() (>0.65)
    - Aumentar ENERGY_THRESHOLD en entornos ruidosos
3. **No reconoce palabras específicas**:
    - Realizar calibración para esa palabra específica
    - Verificar que se pronuncie de manera consistente y similar a la calibración
4. **LEDs no se encienden**:
    - Verificar conexiones de los LEDs y resistencias
    - Comprobar definición de pines en el código

Recomendaciones de Uso

1. **Entorno de Uso**:
    - El sistema funciona mejor en entornos con ruido ambiental moderado a bajo

    - Hablar claramente y a una distancia consistente del micrófono (15-30 cm)
2. **Opciones de Mejora**:
    - Aumentar NUM_BANDS para mayor resolución de frecuencia
    - Implementar filtrado de ruido o normalización dinámica
    - Ajustar tiempos de respuesta según necesidades
3. **Expansión del Sistema**:
    - Se pueden añadir más comandos modificando el array commands[]
    - Posibilidad de añadir otros actuadores además de LEDs