Reconocimiento de Voz con ESP32 y TensorFlow Lite Micro

Este proyecto implementa un sistema de reconocimiento de voz en tiempo real utilizando un microcontrolador ESP32-WROOM-32 y TensorFlow Lite Micro. El sistema permite entrenar un modelo con comandos de voz personalizados (por defecto: "adelante", "atras", "derecha", "izquierda" y "ruido") para controlar cuatro LEDs conectados a pines GPIO. El proyecto está optimizado para microcontroladores con recursos limitados y utiliza un modo de sueño profundo para reducir el consumo de energía.

Tabla de Contenidos

Requisitos

Hardware

- **Microcontrolador**: ESP32-WROOM-32
- **Micrófono**: INMP441 (interfaz I2S)
- **LEDs**: 4 LEDs conectados a pines GPIO (12, 13, 14, 15)
- **Resistencias**: 220Ω para limitar la corriente en los LEDs
- **Cables y Protoboard**: Para las conexiones

Software

- **PlatformIO**: Para el desarrollo y carga del código en el ESP32
- **Python 3.x**: Con las bibliotecas tensorflow, librosa, scikit-learn, numpy, sounddevice
- **Git Bash** (en Windows): Para convertir el modelo .tflite a .h
- **Arduino IDE** (opcional): Para monitoreo serial

Estructura del Proyecto

El proyecto está organizado de la siguiente manera:

```
proyecto-voz-esp32/
├── /src                    # Código para el ESP32
│   ├── main.cpp            # Código principal
│   └── modelo_comandos_tflite.h  # Modelo convertido a array de bytes
├── /scripts                # Scripts de Python
│   ├── audios.py           # Grabación de audios
│   ├── RedNeuronal_Español.py  # Entrenamiento y exportación del modelo
│   └── convertir_modelo.py  # Script opcional para convertir .tflite a .h
├── /docker                 # Archivos de Docker (si se usa)
│   ├── Dockerfile
│   └── ...
├── /docs                   # Documentación
│   └── README.md           # Este archivo
├── /data                   # Dataset (no subir a GitHub si es grande)
│   ├── /adelante
│   ├── /atras
│   └── ...
├── .gitignore              # Archivos a ignorar en Git
└── platformio.ini          # Configuración de PlatformIO
```

Instrucciones de Uso

Grabación de Audios

Objetivo: Grabar muestras de audio para entrenar el modelo.

Instalar Dependencias:

```
pip install sounddevice soundfile numpy
```

1.
2. **Configurar audios.py**:
    - Define las palabras en PALABRAS = ["adelante", "atras", "izquierda", "derecha", "ruido"].
    - Establece MUESTRAS_POR_PALABRA = 20.
3. **Grabar Audios**:

Ejecuta:
python scripts/audios.py

-
- Pronuncia cada palabra 20 veces según las instrucciones en pantalla.
- Los audios se guardan en /data/palabra/palabra_i.wav.

Nota: Graba en un entorno silencioso. Para "ruido", usa silencio o sonidos ambientales suaves.

Entrenamiento del Modelo

Objetivo: Crear un modelo TensorFlow Lite Micro con los audios grabados.

Instalar Dependencias:

pip install tensorflow librosa scikit-learn numpy

1.
2. **Configurar RedNeuronal_Español.py**:
    - Asegúrate de que CLASSES coincida con las palabras grabadas.
3. **Entrenar**:

Ejecuta:
python scripts/RedNeuronal_Español.py --dataset_path data --model_path modelo/modelo_comandos.tflite

- El modelo se guarda como modelo_comandos.tflite (<30 KB).

Nota: Si la precisión es baja (<90%), graba más audios o ajusta el script.

Implementación en el ESP32

Objetivo: Cargar el modelo y el código en el ESP32.

1. **Convertir el Modelo**:

Usa Git Bash para convertir el modelo a un array de bytes:
xxd -i modelo/modelo_comandos.tflite > src/modelo_comandos_tflite.h

2. **Configurar main.cpp**:
    - Incluye el modelo con #include "modelo_comandos_tflite.h".
    - Verifica que los pines coincidan con tu hardware.
3. **Compilar y Cargar**:
    - Usa PlatformIO para compilar y cargar el código.
    - Monitorea el serial a 115200 baudios.

Pruebas y Uso

Objetivo: Validar el sistema.

1. **Probar**:
    - Di una palabra (e.g., "adelante") y verifica que el LED correspondiente se encienda.
    - Revisa las predicciones en el monitor serial.
2. **Modo de Sueño**:
    - Sin actividad por 5 segundos, el ESP32 entra en sueño profundo y despierta tras 5 segundos.

Nota: Si falla, revisa audios, conexiones o reentrena el modelo.

Reentrenamiento con Nuevas Palabras

1. **Grabar Nuevos Audios**:
    - Actualiza PALABRAS en audios.py y graba.
2. **Reentrenar**:
    - Ajusta CLASSES en RedNeuronal_Español.py y ejecuta el script.
3. **Actualizar ESP32**:
    - Convierte el nuevo modelo y ajusta main.cpp si cambian las clases.

Consideraciones

- **Formato de Audios**: WAV, 16kHz, 1 segundo, mono.
- **Tamaño del Modelo**: <30KB. Simplifica la CNN si es necesario.
- **Pines**: Ajustables en main.cpp.
- **Ruido**: Entrena la clase "ruido" para evitar falsos positivos.

Soporte

- **Correo**: mecatronico26.eg@gmail.com
- **Teléfono**: +5493875797847

Licencia

Este proyecto está licenciado bajo la Licencia MIT.

Este README.md proporciona una guía completa para entender, replicar y modificar el proyecto. Si tienes alguna pregunta o necesitas ayuda adicional, no dudes en contactarme. ¡Espero que disfrutes trabajando en este proyecto tanto como yo disfruté creándolo!