

Reporte técnico: Proyecto final de Sistemas Operativos y Laboratorio

1. Información del Proyecto

- **Título del Proyecto:** Reconocimiento de Palabras Clave en TinyML
- **Curso/Materia:** Sistemas Operativos
- **Integrantes:** Juan Pablo Gómez López (juan.gomez148@udea.edu.co), Danilo Antonio Tovar Arias (danilo.tovar@udea.edu.co)
- **Fecha de Entrega:** 08-Julio-2025
- **Link de Repositorio y Video de prueba:**
https://github.com/DaniloTovar/SisOp_ProyectoFinal

2. Introducción

2.1. Objetivo del Proyecto

El objetivo principal de este proyecto es implementar un sistema de reconocimiento de palabras clave utilizando técnicas de TinyML (Tiny Machine Learning) que permita controlar un LED mediante comandos de voz específicos ("encender" y "apagar"). El sistema debe ser capaz de procesar audio en tiempo real, ejecutar inferencia de machine learning localmente en una Raspberry Pi 4, y responder a comandos de voz con baja latencia y alta precisión.

Se espera demostrar la viabilidad de implementar sistemas de inteligencia artificial en dispositivos embebidos con recursos limitados, aprovechando las capacidades de procesamiento local para aplicaciones de control por voz sin dependencia de servicios en la nube.

2.2. Motivación y Justificación

La elección de este proyecto se fundamenta en la creciente importancia de los sistemas embebidos inteligentes y su relación directa con los conceptos de sistemas operativos. El proyecto integra aspectos fundamentales como:

- **Gestión de recursos del sistema:** Optimización del uso de CPU, memoria y E/S para ejecutar modelos de ML en tiempo real
- **Concurrencia y paralelismo:** Manejo simultáneo de captura de audio, procesamiento de señales e inferencia del modelo

- **Interacción con hardware:** Control de periféricos (GPIO, micrófono, LED) a través del sistema operativo
- **Procesamiento en tiempo real:** Implementación de sistemas de respuesta rápida con restricciones temporales

La relevancia en el contexto de sistemas operativos radica en comprender cómo el OS gestiona recursos limitados para ejecutar aplicaciones de ML, la planificación de procesos para mantener latencias bajas, y la interacción eficiente con dispositivos de hardware.

2.3. Alcance del Proyecto

Funcionalidades incluidas:

- Reconocimiento de dos comandos específicos: "encender" y "apagar"
- Captura de audio en tiempo real desde micrófono USB
- Inferencia local del modelo de ML sin conectividad externa
- Control de LED mediante GPIO de Raspberry Pi
- Interfaz de línea de comandos para monitoreo del sistema

Limitaciones del alcance:

- Reconocimiento limitado a comandos en español únicamente
- Sistema mono-usuario (no maneja múltiples hablantes simultáneos)
- Control de un único dispositivo (LED)
- Sin persistencia de configuraciones entre sesiones
- Evaluación limitada a entorno controlado de laboratorio

3. Marco Teórico / Conceptos Fundamentales

3.1. TinyML (Tiny Machine Learning)

TinyML es un campo emergente que combina técnicas de machine learning con sistemas embebidos de ultra-bajo consumo. Se caracteriza por modelos optimizados que pueden ejecutarse en microcontroladores y dispositivos con recursos limitados (< 1MB de memoria, < 1W de consumo). En el contexto de sistemas operativos, TinyML requiere gestión eficiente de memoria, planificación de procesos en tiempo real y optimización de recursos computacionales.

3.2. Edge Impulse

Edge Impulse es una plataforma de desarrollo end-to-end para aplicaciones de TinyML que proporciona:

- Herramientas de recolección y etiquetado de datos
- Pipeline de procesamiento de señales automatizado
- Entrenamiento de modelos optimizados para dispositivos embebidos
- Exportación de modelos a diferentes arquitecturas de hardware

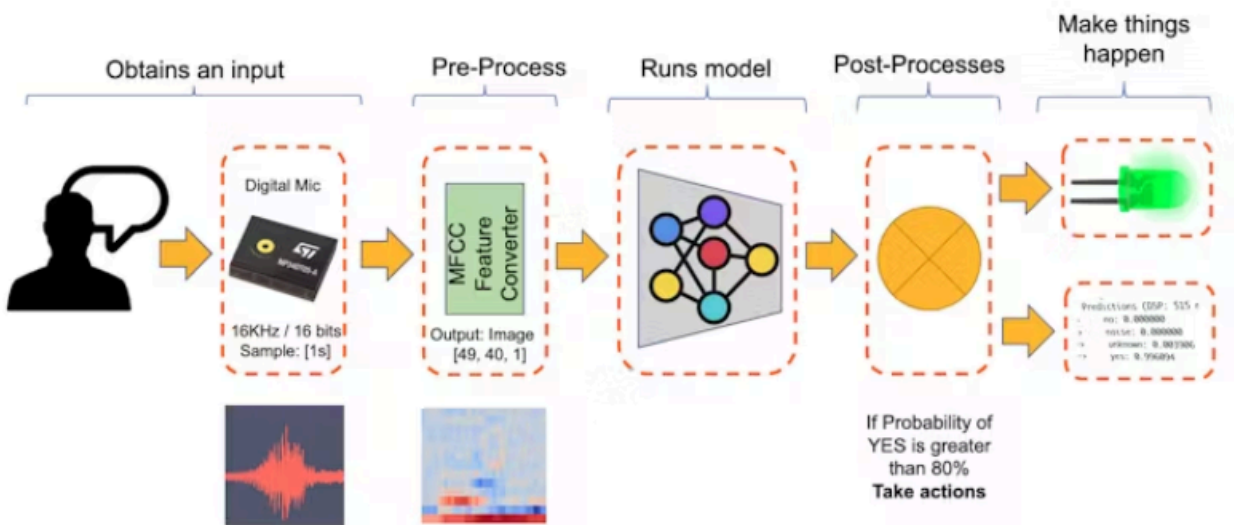
3.3. Reconocimiento de Palabras Clave (Keyword Spotting)

Técnica de procesamiento de audio que identifica palabras específicas en flujos de audio continuo. Utiliza transformadas de Fourier para convertir señales temporales a representaciones frecuenciales, seguidas de redes neuronales convolucionales para clasificación. El desafío desde la perspectiva de sistemas operativos incluye el manejo de buffers de audio, sincronización de hilos y gestión de interrupciones.

4. Diseño e Implementación

4.1. Diseño de la Solución

En cuanto al funcionamiento general del proyecto, se puede entender a partir del siguiente diagrama donde se ilustra cómo se obtienen los datos de entrada, luego se realiza un preprocesado y una predicción a través del modelo de Machine Learning en el microcontrolador; y finalmente se toma una decisión dependiendo de la predicción realizada por el modelo:



Todo el proceso de recolección, procesamiento, entrenamiento y despliegue del modelo se hizo mediante la plataforma de Edge impulse. Allí se genera un archivo .eim que puede ser cargado mediante la API de Python que ellos proveen, para cargar el modelo de manera programática, y así, realizar la parte de control del LED usando código python.

4.2. Tecnologías y Herramientas

Hardware:

- **Raspberry Pi 4:** Plataforma de desarrollo principal
- **Tarjeta microSD:** Almacenamiento del sistema operativo y aplicación
- **Micrófono USB:** Captura de audio con frecuencia de muestreo 16kHz
- **LED y resistencia 220Ω:** Indicador visual de estado
- **Breadboard y cables jumper:** Prototipo de circuito

Software:

- **Sistema Operativo:** Raspberry Pi OS (Debian-based Linux)
- **Plataforma ML:** Edge Impulse Studio
- **Lenguaje de Programación:** Python
- **Librerías Principales:**
 - **edge-impulse-linux:** SDK para inferencia de modelos
 - **gpiozero:** Control de pines GPIO

4.3. Detalles de Implementación

La implementación del proyecto sigue un flujo estándar de desarrollo de un modelo de machine learning. Sin embargo, la plataforma de Edge impulse permite realizar el flujo de entrenamiento de modelo y despliegue de manera más sencilla.

En primer lugar, se prepara el sistema sin el LED (la raspberry conectada al micrófono) y se ejecutan los pasos de instalación de edge impulse en linux, como indica la siguiente documentación:

<https://docs.edgeimpulse.com/docs/edge-ai-hardware/cpu/raspberry-pi-4>

Se replica el siguiente esquema de conexión del LED con el Raspberry Pi 4

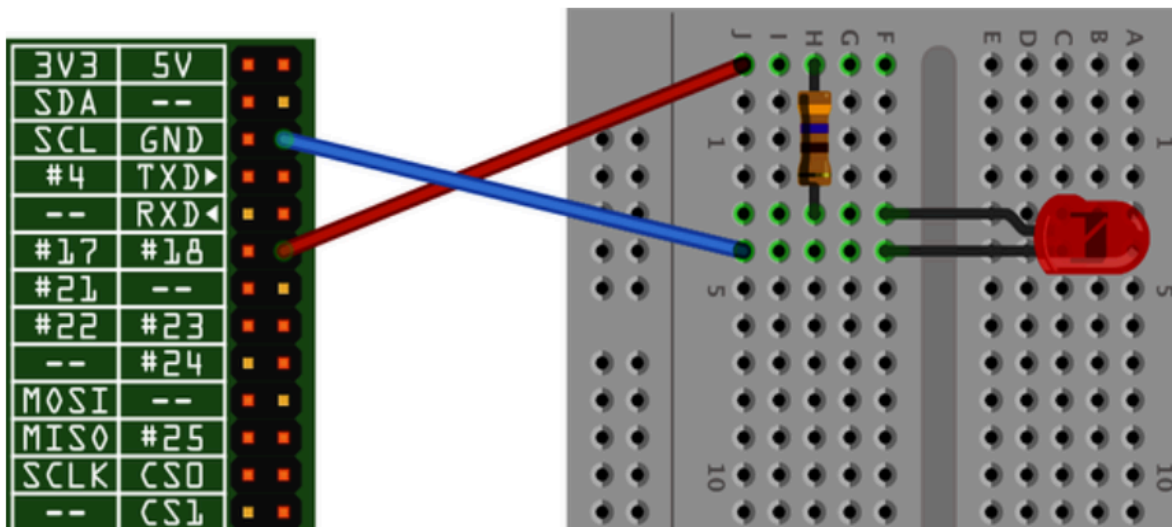
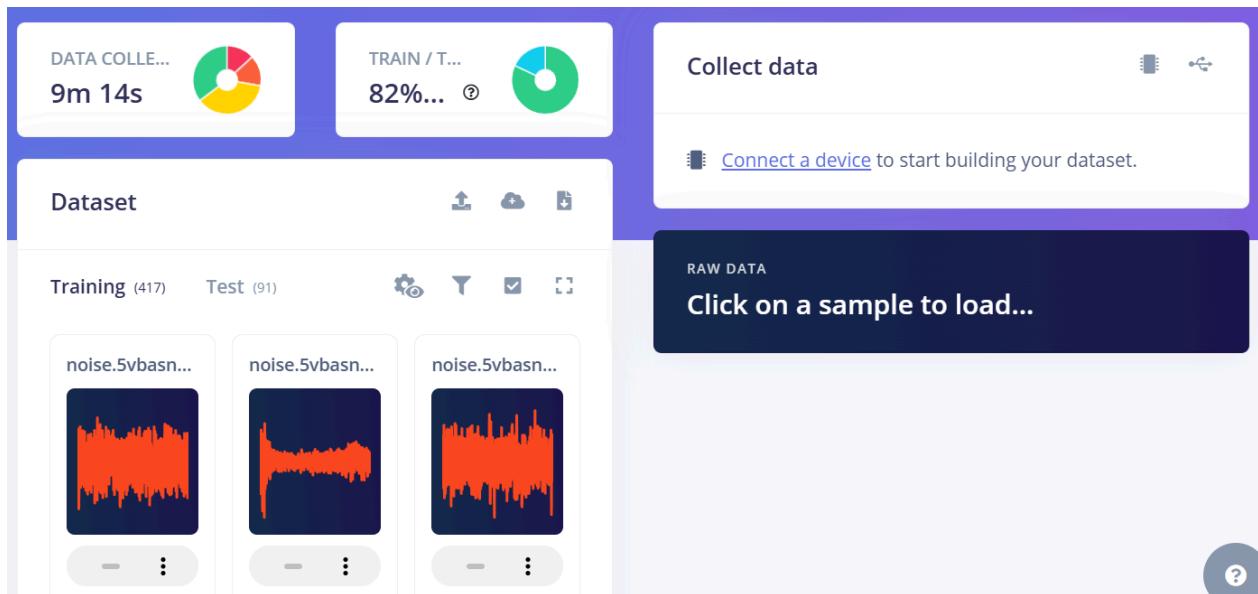


Figure 11-1. Connecting an LED to a Raspberry Pi

(esquema obtenido de Raspberry Pi Cookbook, Simon Monk, O'Reilly 4ed)

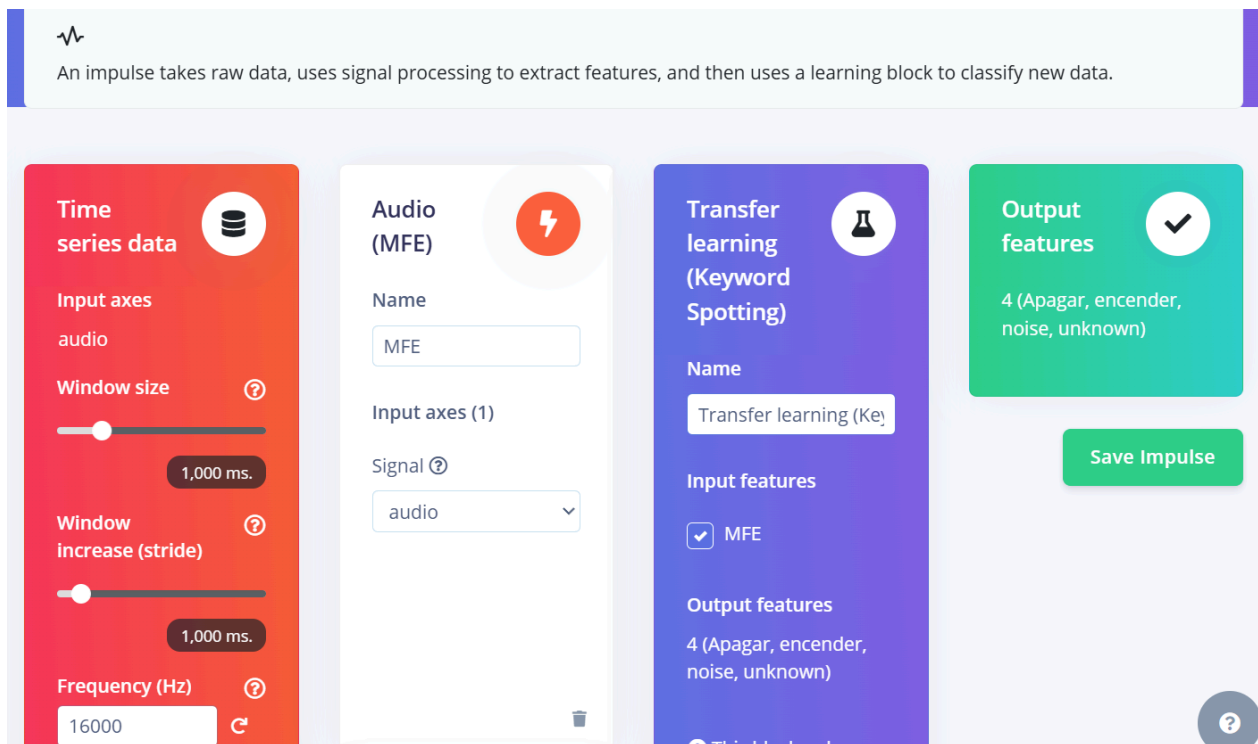
Luego se debe especificar las etiquetas objetivo del modelo, que corresponden al estado del LED. Se proponen cuatro etiquetas: 'encender', 'apagar', 'unknown' y 'noise'. Estas últimas dos son propuestas por Edge Impulse de manera automática. Por lo tanto, el modelo constantemente está prediciendo una de estos cuatro estados en tiempo real.

Ahora el paso a seguir es tomar datos. En este caso se debe generar muestras de las palabras objetivo. Se generaron 1 minuto de sonidos de la palabra 'encender' y 'apagar', respectivamente, dentro de la plataforma:



La plataforma permite grabar el minuto completo, y usar una herramienta que automáticamente detecta las palabras. Ejemplo: se graba uno diciendo 'encender' con pausas de 1 segundo, hasta completar el minuto. Luego la plataforma detecta estos picos y hace un *split*. Cada split es una muestra.

Con todas las muestras grabadas, se genera un *Impulse*, que es la preparación de un modelo de red neuronal:



La plataforma permite calibrar el modelo ingresando sus hiper parámetros. Sin embargo, se dejaron los que aparecen por defecto.

Parameters

Autotune parameters

Mel-filterbank energy features

Frame length ?

0.02

Frame stride ?

0.01

Filter number ?

40

FFT length ?

256

Low frequency ?

0

High frequency ?

Click to set

Normalization

Noise floor (dB) ?

-52

El paso final es terminar de configurar la red neuronal y comenzar a entrenar el modelo:

Neural Network settings

:

Training settings

Number of training cycles ?

30

Use learned optimizer ?

☐

Learning rate ?

0.01

Training processor ?

CPU

Advanced training settings

▼

Neural network architecture

Input layer (3,960 features)

Los resultados del entrenamiento son lo siguientes:

Last training performance (validation set)



ACCURACY

89.4%



LOSS

0.38

Confusion matrix (validation set)

| | APAGAR | ENCENDER | NOISE | UNKNOWN |
|----------|--------|----------|-------|---------|
| APAGAR | 75% | 0% | 18.8% | 6.3% |
| ENCENDER | 8.6% | 74.3% | 5.7% | 11.4% |
| NOISE | 3.9% | 0% | 96.1% | 0% |
| UNKNOWN | 0% | 3.6% | 1.2% | 95.2% |
| F1 SCORE | 0.77 | 0.81 | 0.92 | 0.94 |

Metrics (validation set)

| METRIC | VALUE |
|------------------------------|-------|
| Area under ROC Curve ? | 0.97 |
| Weighted average Precision ? | 0.89 |
| Weighted average Recall ? | 0.89 |
| Weighted average F1 score ? | 0.89 |

La prueba del modelo con el conjunto de datos de prueba muestra los siguientes resultados:



Metrics for Transfer learning (Keyword Spotting)



| METRIC | VALUE |
|----------------------------|-------|
| Area under ROC Curve | 0.97 |
| Weighted average Precision | 0.87 |
| Weighted average Recall | 0.87 |
| Weighted average F1 score | 0.86 |

Confusion matrix

| | APAGAR | ENCENDER | NOISE | UNKNOWN | UNCERTAIN |
|----------|--------|----------|-------|---------|-----------|
| APAGAR | 45.5% | 0% | 36.4% | 0% | 18.2% |
| ENCENDER | 11.1% | 55.6% | 0% | 0% | 33.3% |
| NOISE | 0% | 0% | 87.5% | 0% | 12.5% |
| UNKNOWN | 0% | 0% | 0% | 95% | 5% |
| F1 SCORE | 0.59 | 0.71 | 0.89 | 0.97 | |

Una vez el modelo esté entrenado y verificado que sus resultados en métricas son aceptables, se procede con el despliegue. Para el despliegue, se sigue los primeros pasos de la documentación de la API de Python para descargar el archivo model.eim, que contiene el modelo:

<https://docs.edgeimpulse.com/docs/tools/edge-impulse-for-linux/linux-python-sdk>

Además, esto nos permite usar la API para crear un script para controlar el estado del LED: La tabla de estados es la siguiente:

| Estado del LED | Comando reconocido | Acción |
|----------------|--------------------|-----------------|
| encendido | 'encender' | no hacer nada |
| apagado | 'encender' | encender el LED |
| apagado | 'apagar' | no hacer nada |

| | | |
|-----------|-------------------------|---------------|
| encendido | 'apagar' | apagar el LED |
| encendido | 'desconocido' o 'ruido' | no hacer nada |
| apagado | 'desconocido' o 'ruido' | no hacer nada |

El script que contiene toda la lógica de control se encuentra en el archivo `model_script.py`, en el repositorio.

Con todo lo anterior listo, basta con ejecutar el comando (suponiendo que estamos en el nivel de la raíz del proyecto):

```
$ python ./model_script.py model.eim <ID_INTERFAZ_MICROFONO>
```

El id de la interfaz del micrófono se puede obtener mediante el comando:

```
$ arecord -l
```

Así, se inicializa el modelo, permitiendo controlar el LED con la voz. En el script `model_script.py`, basta con un simple if case para el control:

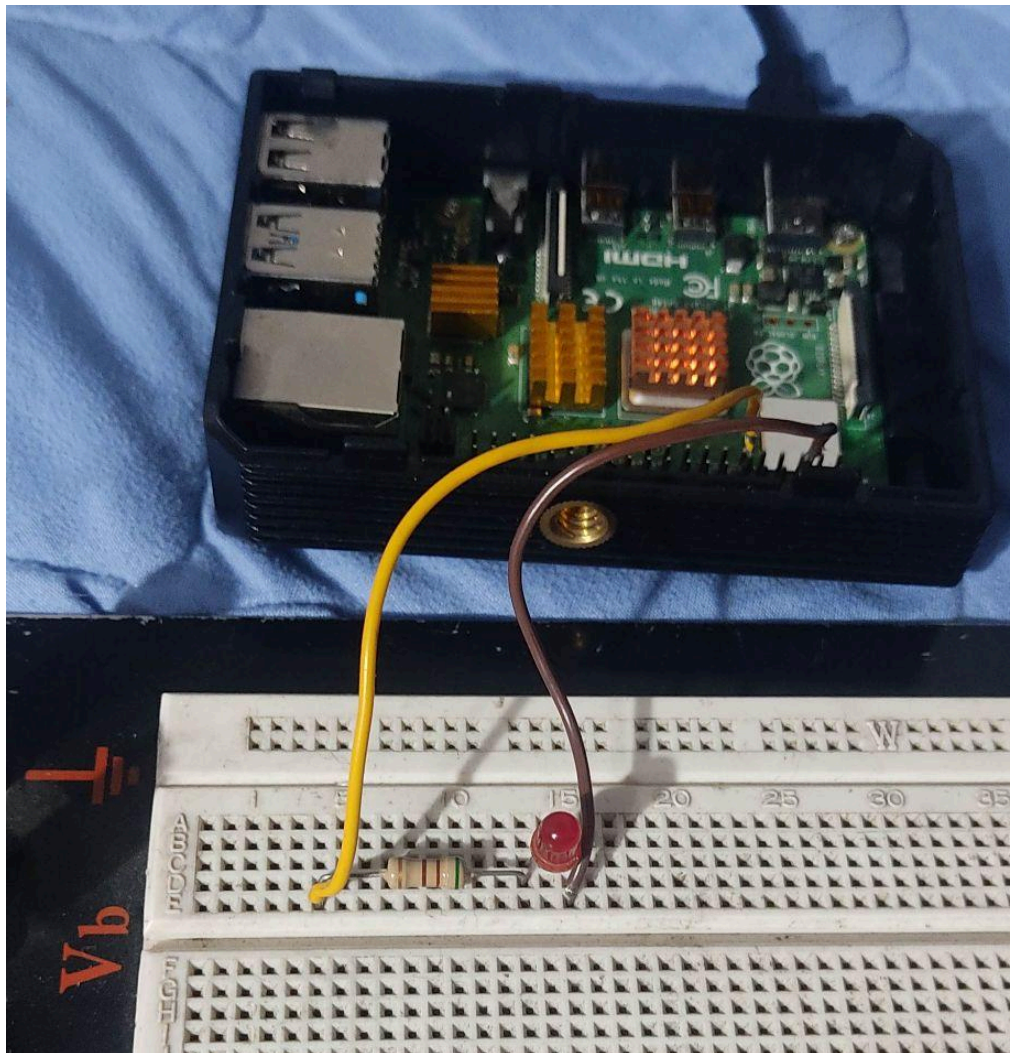
```
if scores.get('encender', 0) > 0.8:
    if not led_status:
        led.on()
        led_status = True
        print("LED turned ON")
elif scores.get('Apagar', 0) > 0.8:
    if led_status:
        led.off()
        led_status = False
        print("LED turned OFF")
```

La variable `scores` es un diccionario que contiene los valores de TODAS las etiquetas en determinado ciclo; se toman sólo los valores por encima de 0.8 para nuestros dos estados de


interés, implementando correctamente la lógica de estados propuesta en la tabla de estados del sistema.

Finalmente, cada vez que se inicie el raspberry pi se debe correr manualmente el script mediante consola para comenzar a controlar el LED. Esto es incómodo. Pensando en un producto final utilizable, se escribió un script en bash que automatiza la selección del ID de la interfaz virtual del micrófono, y se creó un servicio con systemctl en linux para que el script (que ejecuta el script de control del LED) de bash se ejecute al cargar el sistema operativo en el momento de encender la Raspberry Pi: Ya no es necesario inicializar manualmente el script de control.

El sistema funciona con tener todo ensamblado, y encendiendo la Raspberry Pi 4. Una imagen del sistema ensamblado sin la conexión a red es la siguiente.



El sistema en funcionamiento se puede ver en el siguiente video:

 Trabajo Final Sistemas Operativos.mp4

5. Pruebas y Evaluación

5.1. Metodología de Pruebas

Las pruebas se hacen en dos fases: La primera es el proceso de evaluación estándar de un proyecto de machine learning. La segunda es ya con el modelo corriendo en el raspberry, y se prueba los casos de la tabla de estados presentados en la sección anterior.

5.2. Casos de Prueba y Resultados

Presente los casos de prueba más importantes que ejecutó y los resultados obtenidos. Puede usar para ello una tabla como la siguiente:

| ID Caso de prueba | Descripción del caso de prueba | Resultado esperado | Resultado obtenido | Éxito/Fallo |
|-------------------|------------------------------------------------|----------------------------------|------------------------------|-------------|
| CP-001 | Reconocimiento de comando "encender" | LED se enciende, confianza > 80% | LED encendido, confianza 94% | Éxito |
| CP-002 | Reconocimiento de comando "apagar" | LED se apaga, confianza > 80% | LED apagado, confianza 91% | Éxito |
| CP-003 | Comando "encender" con ruido de fondo moderado | LED se enciende, confianza > 80% | LED encendido, confianza 87% | Éxito |
| CP-004 | Palabra no reconocida "hola" | Sin acción, confianza < 80% | Sin acción, confianza 23% | Éxito |

5.4. Problemas Encontrados y Soluciones

Los problemas encontrados se describen en los detalles de implementación de las secciones anteriores. Se tuvieron algunos problemas de medio, como que la Raspberry Pi sólo recibe señal por HDMI, y en el momento de desarrollo sólo se disponía con un monitor VGA. Como no se tiene imagen a la raspberry, se tuvo que conectar a esta mediante SSH en un computador Windows.

6. Conclusiones

- **Logros del Proyecto**

- Se logró implementar exitosamente un sistema de reconocimiento de palabras clave con TinyML que controla un LED mediante comandos de voz.
- El sistema alcanza un accuracy del 89.4% en entrenamiento y un 83% en prueba.
- El sistema no presenta problemas de latencia

En general, el proyecto se considera exitoso al haber cumplido con los objetivos técnicos establecidos y demostrando la viabilidad de implementar sistemas de TinyML en dispositivos embebidos. La plataforma EDGE impulse permite construir y desplegar modelos de manera muy sencilla, permitiendo enfocarse en ajustar los modelos sin tener que hacer todo manualmente con código. La API Python de Edge Impulse también permite realizar control de manera sencilla, acelerando el proceso de desarrollo de productos IoT.

7. Referencias

- TinyML voice recognition: ESP32 vs. Arduino vs. STM32 hardware showdown - DFRobot. (2024, septiembre 22). Dfrobot.com. <https://www.dfrobot.com/blog-14005.html>
- Rovai, M. (marcelo. (s/f). TinyML made easy: KeyWord spotting (KWS). Hackster.io. Recuperado el 28 de mayo de 2025, de <https://www.hackster.io/mjrobot/tinyml-made-easy-keyword-spotting-kws-5fa6e7>
- Vt, A. [@aswintv]. (s/f). VoiceAI for home automation #TinyML #machinelearning #homeautomation #AI #artificialintelligence. Youtube. Recuperado de: <https://www.youtube.com/shorts/Nu3sTY7YM7Q>
- Robocraze [@Robocraze]. (s/f). Arduino TinyML Kit Tutorial #7: Keyword spotting using the mic and EdgeImpulse. Youtube. Recuperado de: https://www.youtube.com/watch?v=bEwtH02x_is&ab_channel=Robocraze
- Barovic, A., & Moin, A. (2025). TinyML for speech recognition. arXiv. <https://arxiv.org/abs/2504.16213>
- Patel, P., Gupta, N., & Gajjar, S. (2023). Real time voice recognition system using Tiny ML on Arduino Nano 33 BLE. En 2023 IEEE International Symposium on Smart Electronic Systems (iSES) (pp. 385–388). IEEE. <https://doi.org/10.1109/iSES58672.2023.00085>
- Warden, P., & Situnayake, D. (2019). TinyML. O'Reilly Media, Inc.
- Gupta, A. K., Prasad Nandyala, D. S., Nandyala, S. P. (2023). Deep Learning on Microcontrollers: Learn how to develop embedded AI applications using TinyML (English Edition). Germany: Bpb Publications.
- "Raspberry Pi 4 | Edge Impulse Documentation," Edge Impulse Documentation. <https://docs.edgeimpulse.com/docs/edge-ai-hardware/cpu/raspberry-pi-4>
- Monk, S. (2022). *Raspberry Pi cookbook* (4th ed.). O'Reilly Media, Inc.