RIELAC, Vol. 45(2):e2402 (2024) ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Algoritmo de cómputo de espectrogramas de Mel en tiempo real en microcontroladores STM32 para la detección de palabras clave en voz

Alejandro Perdomo-Campos, Jorge Ramírez-Beltrán, Arturo Morgado-Estevez

RESUMEN / ***ABSTRACT***

La detección de palabras clave es un subcampo del reconocimiento automático de voz revolucionado en la última década con la incorporación de técnicas de inteligencia artificial basadas en aprendizaje profundo. La implementación de modelos de detección de palabras clave en microcontroladores implica como primer paso el procesamiento digital de las señales de audio para realizar la extracción de características en tiempo real. Los modelos implementados en microcontroladores encontrados en la literatura usan los coeficientes MFCC para la extracción de características. Sin embargo, se ha comprobado que al emplear técnicas de aprendizaje profundo para la clasificación resulta más efectivo el empleo de espectrogramas de Mel. En este artículo se propone una implementación del algoritmo para la obtención de espectrogramas de Mel en tiempo real en microcontroladores de la familia STM32 compatible con el diseño de un sistema de detección de palabras clave en tiempo real, evaluándose su uso en un sistema de prueba basado en el microcontrolador STM32G474RET6 y el micrófono MEMS SPH0645LM4H-B. La implementación propuesta reduce el uso de memoria RAM en el microcontrolador y llena el vacío existente en el driver CMSIS-DSP de una rutina para el cómputo del espectro de Mel de un vector de muestras de señal.

Palabras claves: detección de palabras clave, extracción de características, espectrograma de Mel, STM32

Keyword spotting is a subfield of automatic speech recognition revolutionized in the last decade with the incorporation of artificial intelligence techniques based on deep learning. The implementation of keyword spotting models in microcontrollers implies as a first step the digital processing of audio signals to carry out the feature extraction in real time. The models implemented in microcontrollers found in literature use MFCC coefficients for feature extraction. It has been proved that when using deep learning techniques for classification, it is more effective the use of Mel spectrograms. In this paper, it is proposed an implementation of the algorithm for the obtention of Mel spectrograms in real time in STM32 microcontrollers compatible with the design of a real time keyword spotting system. Its use is evaluated in a test system based on the STM32G474RET6 microcontroller and the MEMS microphone SPH0645LM4H-B. The proposed implementation reduces the use of RAM memory in the microcontroller and fills the existing gap in the CMSIS-DSP driver of a routine for the computation of the Mel spectrum for a vector of signal samples.

Keywords: keyword spotting, feature extraction, Mel spectrogram, STM32

Algorithm for the real-time computation of Mel spectrograms in STM32 microcontrollers for keyword spotting in voice.

Recibido: 04/2024 Aceptado: 07/2024

1. -INTRODUCCIÓN

La detección de palabras clave en señales continuas de voz es un subcampo del reconocimiento automático de voz aplicado al diseño de dispositivos y sistemas activados por voz. En este principio basan su funcionamiento los asistentes inteligentes como Google Assistant, Siri, Alexa o Cortana, tan populares hoy en día [1].

El estado del arte de los modelos de reconocimiento de patrones para aplicaciones de detección de palabras clave ha quedado determinado en la última década por la incorporación de técnicas de inteligencia artificial basadas en aprendizaje profundo. Debido a la complejidad computacional que poseen los algoritmos de aprendizaje profundo, desplegar dichos algoritmos en plataformas limitadas en recursos de *hardware* como son los microcontroladores implica grandes retos. Sin embargo, desde que comenzó el desarrollo práctico de sistemas de este tipo en sistemas embebidos, se ha logrado implementar en microcontroladores algoritmos con niveles de complejidad apropiados para diversas aplicaciones, con más éxito en los últimos años debido al auge de paradigmas tecnológicos como el *TinyML* [2] .

En los modelos de detección de palabras clave que usan algoritmos de aprendizaje profundo como clasificadores, se usa una etapa previa de extracción de características de las señales de audio, de la cual se obtienen parámetros o representaciones descriptivas de las señales en otro dominio que facilitan el reconocimiento de patrones de voz [3] . Es por ello que la implementación de estos modelos en microcontroladores implica como primer paso el procesamiento digital de las señales de audio para realizar la extracción de características en tiempo real.

Existe un gran número de métodos de extracción de características de señales de audio. Alías et al. ofrecen una amplia revisión de los mismos [4]. Por una parte, están los métodos que extraen solo características físicas de la señal como la variabilidad en el tiempo, la tasa de cambio de signo, la periodicidad, la potencia o la amplitud de la señal. Por otro lado, están aquellos métodos que integran la percepción de los sonidos en el proceso de parametrización a través del cálculo de parámetros de la señal que describen aspectos relevantes de la percepción del sonido. Estos métodos perceptuales típicamente incluyen en el proceso de parametrización modelos simplificados del sistema auditivo humano como los de Bark, Mel y Gammatone. Tanto el enfoque físico como el perceptual agrupan una gran variedad de técnicas que operan sobre la señal en diferentes dominios.

De todas las técnicas de extracción de características incluidas en la revisión de [4] , las técnicas basadas en la escala perceptual de Mel están entre las más empleadas en aplicaciones orientadas al reconocimiento automático de voz [5–7] . La mayoría de los modelos de detección de palabras clave basados en técnicas de aprendizaje profundo del estado del arte usan espectrogramas de Mel o los coeficientes cepstrales de frecuencia de Mel (MFCC por sus siglas de *Mel Frequency Cepstral Coefficients*) para la extracción de características [3] . Sin embargo, los trabajos que se encuentran en la literatura donde se implementan modelos de este tipo en microcontroladores usan los coeficientes MFCC debido a que la matriz de coeficientes que se obtiene es de menores dimensiones y por ende más fácil de almacenar en sistemas con limitaciones de memoria.

Ye et al. usaron MFCC para la extracción de características en un microcontrolador Sunplus para la detección de palabras aisladas [8] . Miah y Wang implementaron en [9] un sistema de detección de palabras clave usando MFCC para la extracción de características en un microcontrolador STM32F769NI y en una computadora de placa única Jetson Nano para comparar su desempeño en ambas plataformas. Rusci y Tuytelaars desplegaron varios modelos de detección de palabras clave usando técnicas de personalización al hablante en un microcontrolador GAP9, usando MFCC para la extracción de características [10] . Li et al. combinaron la extracción de los MFCC con una Transformada Discreta de Wavelet para la extracción de características, llegando a validar su propuesta en un microcontrolador ESP32 [11] .

Si bien es cierto que el almacenamiento de espectrogramas de Mel en un microcontrolador implica un mayor costo en memoria, el cálculo de los coeficientes MFCC elimina información y destruye las relaciones temporales de los parámetros contenidos en el espectrograma de Mel. Se ha comprobado que cuando se usan modelos basados en aprendizaje profundo para el reconocimiento de patrones en señales de voz, resulta más apropiado emplear directamente espectrogramas logarítmicos de Mel como entradas [12] .

Otro enfoque que ha sido explorado es el de incluir en los modelos de aprendizaje profundo la obtención del espectrograma de Mel, sustituyendo el algoritmo de preprocesamiento por capas de redes neuronales para la obtención de un modelo *end-to-end* [13] ; pero este enfoque trae como inconveniente que complejiza la estructura de las redes neuronales a emplear, lo cual se traduce en un mayor costo en memoria del modelo.

Los modelos actuales más avanzados de reconocimiento de voz como *Whisper*, recientemente presentado por la empresa *OpenAI*, usan espectrogramas de Mel como entradas a los complejos modelos basados en Transformers que utilizan. Estos modelos realizan la etapa de preprocesamiento de forma independiente [14] . Sin embargo, están diseñados para ser ejecutados en plataformas sin limitaciones de memoria o procesamiento.

Algunas de las familias más populares de microcontroladores consolidadas en aplicaciones de procesamiento digital de señales son STM32 y MSP430. En el caso de los microcontroladores STM32 basados en núcleos ARM Cortex-M, su conjunto de

instrucciones DSP y la presencia de una unidad aritmética de punto flotante (FPU por sus siglas en inglés de *Floating Point Unit*) asociadas a su arquitectura de 32 bits los hacen una poderosa herramienta para realizar tareas de este tipo manipulando datos de alta resolución [15] .

Por otro lado, los microcontroladores MSP430 son bien conocidos por sus bajos índices de consumo de potencia y por poseer un acelerador de bajo consumo que actúa como coprocesador (LEA por sus siglas en inglés de *Low Energy Accelerator*) para la ejecución de operaciones matemáticas de alta complejidad con independencia de la CPU.

Sin embargo, para aplicaciones de audio donde los datos son adquiridos a través de sistemas con interfaces digitales de comunicación I2S o PDM, la familia de los MSP430 resulta menos atractiva por no contar sus microcontroladores con periféricos embebidos para la implementación de estos protocolos por *hardware*.

Otras populares plataformas como ESP32 y Arduino con la incorporación de microcontroladores con procesadores ARM Cortex-M en sus últimas versiones comerciales también se han vuelto comunes, aunque en menor medida, en aplicaciones de este tipo [11] , [16] .

En 2010 la empresa ARM lanzó la primera versión de CMSIS-DSP, un driver para la implementación de rutinas generales de procesamiento digital de señales en sus procesadores Cortex-M y Cortex-A [17] . Las funciones que ofrece el driver para la implementación de estos algoritmos explotan la arquitectura de los procesadores ARM Cortex-M para realizar las operaciones de una forma óptima mediante el empleo de la FPU y su conjunto de instrucciones SIMD.

El driver CMSIS-DSP ofrece soporte para el cálculo de los MFCC mediante la función *arm_mfcc_f32()*, pero no ofrece ninguna función para el cálculo del espectro de Mel de un vector de muestras de señal. Por lo tanto, la obtención del espectro de Mel en estos procesadores implica la implementación de la rutina.

En este artículo se propone una implementación del algoritmo para la obtención de espectrogramas logarítmicos de Mel en tiempo real en microcontroladores de la familia STM32 reduciendo el uso de memoria RAM. El enfoque propuesto se basa en realizar el procesamiento del audio adquirido de forma simultánea a la adquisición. Para ello se propone el uso de la FPU de 32 bits de los procesadores ARM Cortex-M para acelerar los cálculos matemáticos y un controlador de acceso directo a memoria (DMA por sus siglas en inglés de *Direct Memory Access*) de conjunto con una interfaz de comunicación I2S (Inter-Integrated Sound) para adquirir las señales de audio en formato digital. En la implementación se emplearon algunas de las subrutinas ofrecidas por el driver CMSIS-DSP. La estrategia propuesta permite ajustar fácilmente la extensión temporal de los espectrogramas de acuerdo a las necesidades de la aplicación. Para su validación se diseñó un sistema basado en el microcontrolador de propósito general STM32G474RET6 y el micrófono MEMS SPH0645LM4H-B.

La contribución científica fundamental que se hace en este trabajo es una implementación del algoritmo de obtención de espectrogramas de Mel para microcontroladores STM32 que reduce el uso de memoria RAM y es compatible con el diseño de un sistema de detección de palabras clave en tiempo real basado en un microcontrolador de bajos recursos de memoria y procesamiento. Con esta implementación se llena el vacío existente en el driver CMSIS-DSP de una rutina para el cómputo del espectro de Mel de un vector de muestras de señal. Este resultado puede ser extendido a cualquier aplicación de detección de eventos por emisión acústica que emplee espectrogramas como etapa de extracción de características.

En la Sección 2 se explica el proceso de obtención del espectrograma logarítmico de Mel de una señal de audio; en la Sección 3 se describe la estrategia propuesta para la obtención de los espectrogramas de Mel en tiempo real en microcontroladores STM32, mientras que en la Sección 4 se presentan y discuten los resultados obtenidos.

2.- ESPECTROGRAMA LOGARÍTMICO DE MEL DE UNA SEÑAL DE AUDIO

La estructura de un algoritmo de extracción de características para señales de audio puede ser descrita de forma general por medio del diagrama en bloques que se muestra en la Fig. 1 [4].

En la primera etapa, la señal digital de audio pasa por un bloque de pre-énfasis en el que se realiza un proceso de acondicionamiento. Generalmente, en esta etapa se realizan operaciones de filtrado digital para eliminar componentes de frecuencia no deseadas y componentes de ruido que puedan haber sido introducidas en el proceso de adquisición.

En la segunda etapa, la señal es dividida en segmentos conformados por un determinado número de muestras. En este proceso la serie temporal de muestras de señal se convierte en una secuencia continua de bloques de muestras finitos. En la segmentación se emplea un determinado número de muestras de solapamiento para evitar la pérdida de información entre segmentos. La longitud de los segmentos y el número de muestras de solapamiento a emplear depende de la aplicación en

cuestión. De forma general, la longitud de los segmentos debe ser proporcional a la mínima longitud de los eventos acústicos de interés.

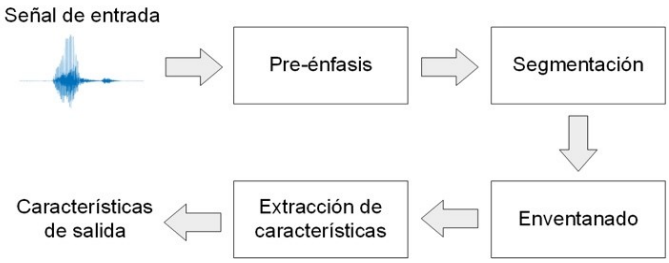


Figura 1

Diagrama en bloques de un algoritmo de extracción de características para señales de audio.

En la tercera etapa se aplica una función de ventana a cada uno de los segmentos con el objetivo de mitigar los efectos del fenómeno conocido como "fugas espectrales" [18] . Las funciones de ventana son aplanadas en los extremos de forma similar a una campana de Gauss, y normalmente se definen con el mismo número de muestras que contienen los segmentos de señal.

En la última etapa se extraen las características acústicas de cada uno de los segmentos de señal enventanados. El objetivo de la extracción de características es obtener una representación compacta de los rasgos más sobresalientes de la señal, convirtiendo el número N de muestras de cada segmento en un número K de coeficientes escalares (con K < N) que brinde información de la señal en otro dominio.

La Fig. 2 muestra en un diagrama en bloques el algoritmo para la obtención del espectrograma logarítmico de Mel de una señal de audio. Los primeros bloques del sistema se corresponden con los bloques generales de los algoritmos de extracción de características ilustrados en la Fig. 1.

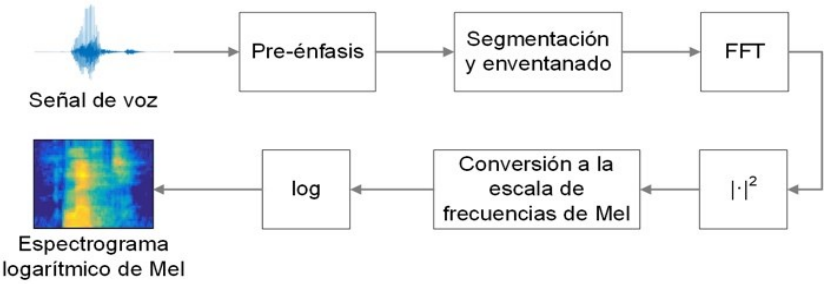


Figura 2

Diagrama en bloques del algoritmo de obtención del espectrograma logarítmico de Mel [3].

El bloque de pre-énfasis normalmente se implementa aplicando un filtro pasa-alto a la señal [19]. Para la segmentación de la señal en el segundo bloque se deben tener en cuenta las características de las señales de voz. Para lograr características acústicas estables, la señal de voz debe ser examinada en períodos de tiempo lo suficientemente cortos. Es común el empleo de segmentos de alrededor de 20 o 30 ms de duración con un solapamiento de un 50%, a los cuales les es aplicada la función de ventana. Las ventanas más comúnmente empleadas en el cálculo de espectrogramas de Mel son las ventanas de Hann y de Hamming que se construyen a partir de funciones coseno. Las ecuaciones (1) y (2) muestran la expresión matemática de las funciones de ventana de Hann y Hamming respectivamente.

$$w_{hann}(n) = 0.5 - 0.5\,cos\left(\frac{2\pi n}{N+1}\right) \quad (1)$$

$$w_{hamming}(n) = 0.46 - 0.46\,cos\left(\frac{2\pi n}{N+1}\right) \quad (2)$$

El tercer y cuarto bloques representan las operaciones necesarias para la obtención de un espectrograma de potencia de la señal. La primera operación es el cálculo de la Transformada Discreta de Fourier (DFT por sus siglas en inglés de *Discrete Fourier Transform*) de cada uno de los segmentos de señal. La ecuación (3) muestra la definición de la DFT, donde x(n) es una señal discreta en el tiempo y N es el número de muestras de la señal digital.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi j n k}{N}} \qquad k = 1,2,3, \dots, N-1 \tag{3}$$

La DFT se calcula por métodos computacionales empleando el algoritmo de la Transformada Rápida de Fourier (FFT por sus siglas en inglés de *Fast Fourier Transform*). Obteniendo las DFT de cada segmento de señal, se obtiene una representación bidimensional tiempo-frecuencia de la señal conocida como espectrograma de amplitud. Luego, se halla el cuadrado del valor absoluto de cada término de las DFT calculadas para obtener la distribución espectral de potencia de la señal en el tiempo o espectrograma de potencia.

El quinto bloque realiza una transformación de los espectros de potencia obtenidos a través de un banco de filtros de Mel. Los filtros de Mel que componen el banco son filtros con función transferencial triangular, de forma que la función transferencial $H_m(k)$ del filtro m-ésimo puede definirse a través de la ecuación (4), donde $f(m)$ es la frecuencia central del filtro triangular m.

$$H_m(k) = \begin{cases} 0 & si\ k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & si\ f(m-1) \le k < f(m) \\ 1 & si\ k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & si\ f(m) < k \le f(m+1) \\ 1 & si\ k > f(m+1) \end{cases} \tag{4}$$

Las ecuaciones (5) y (6) expresan las relaciones entre la escala lineal de frecuencias y la escala de frecuencias de Mel.

$$m = 2595 \log(1 + \frac{f}{700}) \tag{5}$$

$$f = 700\ (10^{\frac{m}{2595}} - 1) \tag{6}$$

El conjunto de segmentos de señal transformados mediante las operaciones descritas hasta este punto representa el espectrograma de Mel. El espectrograma logarítmico puede obtenerse transformando el espectrograma de Mel original a escala logarítmica, como se ilustra en la Fig. 2.

3.- IMPLEMENTACIÓN DEL ALGORITMO DE OBTENCIÓN DEL ESPECTROGRAMA DE MEL EN UN MICROCONTROLADOR STM32

En la Fig. 3 se presenta un diagrama en bloques del *hardware* propuesto para la adquisición y el procesamiento de audio en tiempo real. Se propone emplear como transductor el micrófono MEMS SPH0645LM4H-B de salida digital I2S, del fabricante Knowles [20] . Este micrófono ofrece una alta relación señal-ruido de 65 dB y respuesta en frecuencia plana entre los 100 Hz y 3 kHz, por lo que es adecuado para adquirir señales de voz.

El SPH0645LM4H-B opera como esclavo transmisor en el bus I2S. La señal WS que provee el master en el bus debe tener una frecuencia equivalente a $f_{CLK}/64$, y debe estar sincronizada con la señal CLK. Las frecuencias de reloj f_{CLK} soportadas van desde los 1024 kHz hasta los 4096 kHz, con frecuencias de muestreo equivalentes desde 16 kHz hasta 64 kHz. Se propone el uso de una frecuencia de muestreo de 16 kHz, suficiente para cubrir el ancho de banda de la voz humana con la mínima demanda de recursos en el microcontrolador. El formato de salida de los datos es el I2S estándar, en complemento a 2 con resolución de 24 bits. Las muestras de 24 bits son transmitidas en paquetes de 32 bits.

El SPH0645LM4H-B está diseñado para ser usado en pares para la adquisición de dos canales simultáneos de audio, por lo que al hacer uso de un único canal se deben despreciar los datos que se adquieran en el espacio de tiempo correspondiente al canal que no está siendo utilizado.

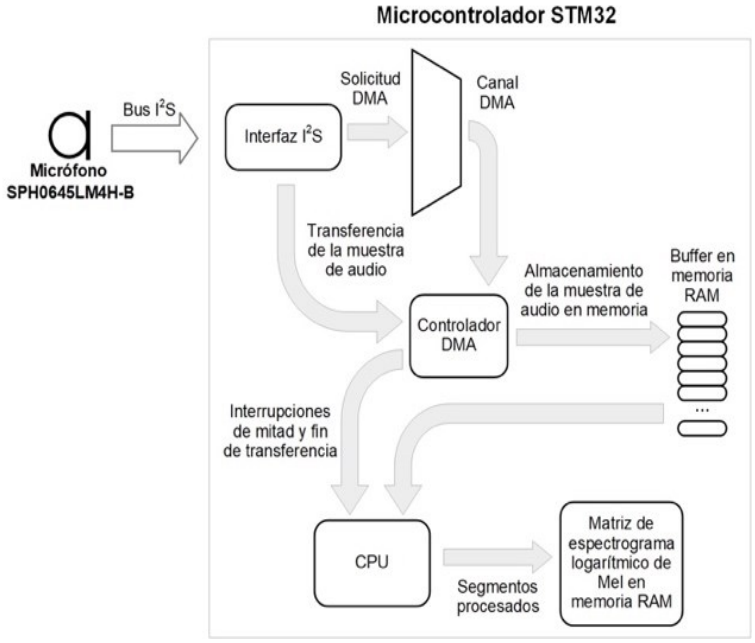


Figura 3

Diagrama en bloques del *hardware* propuesto.

Para la adquisición de los datos en el microcontrolador se propone el uso de una de las interfaces I2S con que cuentan los microcontroladores STM32 en configuración de master receptor. También es posible utilizar los periféricos SAI (*Serial Audio Interface*) con que cuentan algunos STM32 para implementar protocolos de transmisión de audio digital, entre ellos el I2S. Al emplear una de las interfaces I2S se debe usar para la recepción de los datos el registro de 16 bits SPIx_DR asociado, lo cual implica que para la adquisición de cada muestra de audio enviada por el micrófono en 32 bits se requieren dos operaciones de lectura.

Los datos adquiridos por I2S son transferidos a través de un canal DMA a un buffer en memoria RAM. Los datos de audio almacenados en RAM son procesados por la CPU en segmentos de 30 ms para la obtención del espectrograma logarítmico de Mel en tiempo real, el cual es almacenado secuencialmente segmento a segmento en una matriz en memoria RAM que va creciendo de forma dinámica. El uso del DMA en la adquisición de las señales permite a la CPU realizar el procesamiento de un segmento de audio adquirido mientras se está adquiriendo un nuevo segmento, con lo cual se logra la obtención de los espectrogramas sin necesidad de almacenar toda la señal a procesar.

En la Fig. 4 se muestra un diagrama de flujo que describe el ciclo de programa propuesto para la adquisición de audio y la obtención del espectrograma logarítmico de Mel en tiempo real.

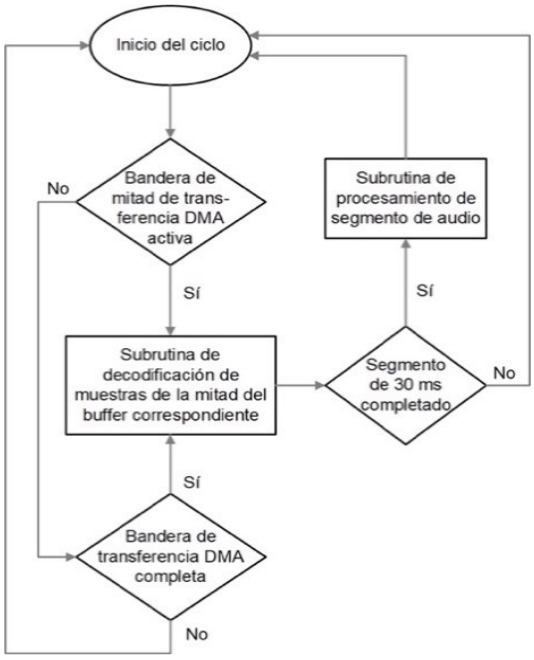


Figura 4

Diagrama de flujo del ciclo de programa principal.

El ciclo se compone de varias subrutinas de programa en las cuales se realizan las operaciones de conformación de las muestras de audio a partir de los datos enviados por el micrófono, la conformación de los segmentos de audio con solapamiento a procesar y el procesamiento de los segmentos conformados. Se emplean segmentos de 30 ms con intervalos de solapamiento de 15 ms entre segmentos. Para la ejecución de estas subrutinas de forma coherente se emplean las interrupciones generadas por el controlador DMA.

3.1.- CONFIGURACIÓN DE LAS TRANSFERENCIAS DMA Y MANEJO DE LA MEMORIA

Las transferencias DMA deben ser configuradas por firmware a nivel de canal DMA. Todas las transferencias se configuran a nivel de bloques de datos, y están compuestas por una secuencia de ciclos de lectura y escritura a través del bus AHB del microcontrolador. Una transferencia en bloque consistirá en una secuencia repetida de:

1. Una operación de lectura a la dirección en memoria de origen.

2. Una operación de escritura a la dirección en memoria de destino.

3. Decremento del valor del registro DMA_CNDTRx que indica el número de operaciones de lectura y escritura a realizar.

Esta secuencia de operaciones se repite hasta que el valor del registro DMA_CNDTRx sea nulo. Durante las transferencias los registros DMA_CPARx y DMA_CMARx que establecen las direcciones de origen y destino mantienen el valor establecido inicialmente.

Para el manejo de la memoria se propone usar el modo circular del DMA. En este modo, al terminar la operación de escritura en la última dirección de destino, el registro DMA_CNDTRx es recargado automáticamente con el valor inicial, y los punteros a las direcciones de lectura y escritura son recargados con las direcciones de base establecidas en los registros DMA_CPARx y DMA_CMARx.

Para la adquisición de los datos de la interfaz I2S, las transferencias DMA a la memoria RAM se deben realizar usando como dirección de origen la dirección del registro SPIx_DR, y como direcciones de destino las direcciones de un buffer habilitado en memoria RAM con capacidad para el almacenamiento de 480 muestras de audio. Esto corresponde a 30 ms de grabación a una frecuencia de muestreo de 16 kHz, dado que este será el tamaño de ventana a utilizar en el cálculo del espectrograma de Mel. Dado que cada muestra de audio equivale a dos lecturas de 16 bits del registro SPIx_DR asociado a la interfaz I2S, y además se deben contemplar las muestras del canal de audio no utilizado, para almacenar 480 muestras el buffer debe

comprender 1920 locaciones de 16 bits (1920=480× 2 × 2.). Se deben usar transferencias de 16 bits y solo habilitarse el incremento del puntero a la dirección de destino, dado que los datos de origen deben ser leídos siempre de la dirección correspondiente al registro SPIx_DR. La Fig. 5 muestra la forma en que se realiza el manejo del buffer circular en memoria usando el DMA, almacenando los datos en el buffer de forma consecutiva.

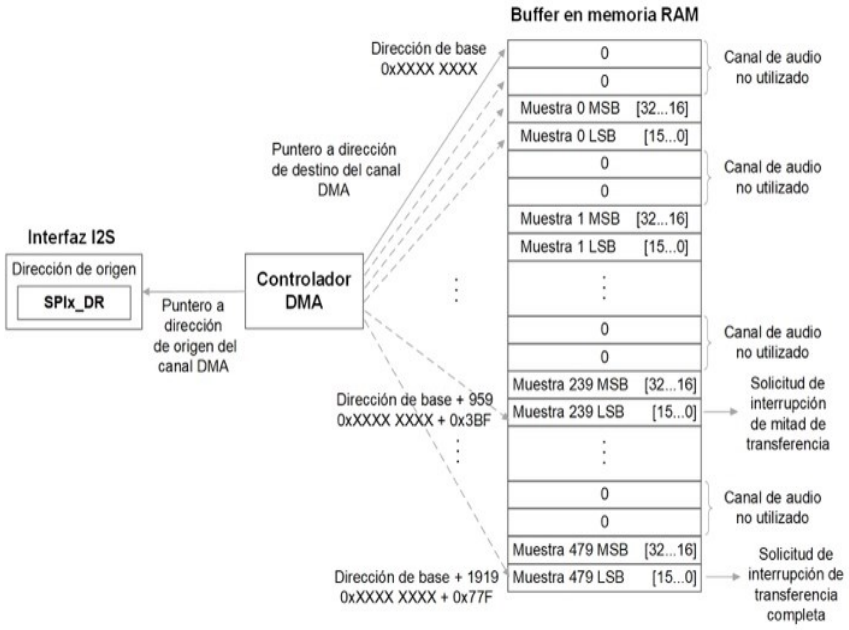


Figura 5

Manejo del buffer circular en memoria RAM por el controlador DMA.

El controlador DMA cuenta con banderas de solicitud de interrupción asociadas a cada canal en el registro DMA_ISR que pueden usarse para generar interrupciones al flujo de programa principal ante la ocurrencia de eventos asociados a la transferencia en curso. Las banderas HTIFx y TCIFx pueden generar interrupciones respectivamente cuando se termina de realizar la mitad de una transferencia o una transferencia completa. Todas las interrupciones provenientes del canal usado son mapeadas al controlador de interrupciones de la CPU a través de la señal de solicitud DMAx_Channelx_IRQn, y vectorizadas a una misma dirección en memoria de programa.

Estas solicitudes de interrupción son usadas por la CPU para el procesamiento de los datos almacenados en el buffer, de forma tal que la CPU pueda procesar los datos de una mitad del buffer mientras la otra mitad del buffer se llena con los datos provenientes de la interfaz I2S. Además, cada mitad del buffer corresponde a 240 muestras de audio, equivalentes a 15 ms de grabación, lo cual facilita la realización del solapamiento necesario entre las ventanas de audio de 30 ms. La Fig. 5. también ilustra la activación de estas solicitudes de interrupción.

3.2.- SUBRUTINA DE DECODIFICACIÓN DE MUESTRAS DE AUDIO

Cada vez que se completa el llenado de una mitad del buffer habilitado en memoria RAM, se ejecuta una subrutina para decodificar las muestras almacenadas en esta mitad del buffer y almacenarlas en el vector del segmento de audio a procesar. Esta subrutina implementa un ciclo para iterar sobre cada par de posiciones en la mitad del buffer y decodificar las muestras de audio, teniendo en cuenta que cada muestra de audio se almacena en forma de dos valores de 16 bits. Esta iteración sobre las posiciones del buffer se realiza descartando los pares de valores de 16 bits correspondientes al canal de audio no empleado.

La Fig. 6 muestra un diagrama de flujo de la subrutina de decodificación de una muestra de audio almacenada en el buffer como dos valores consecutivos de 16 bits.

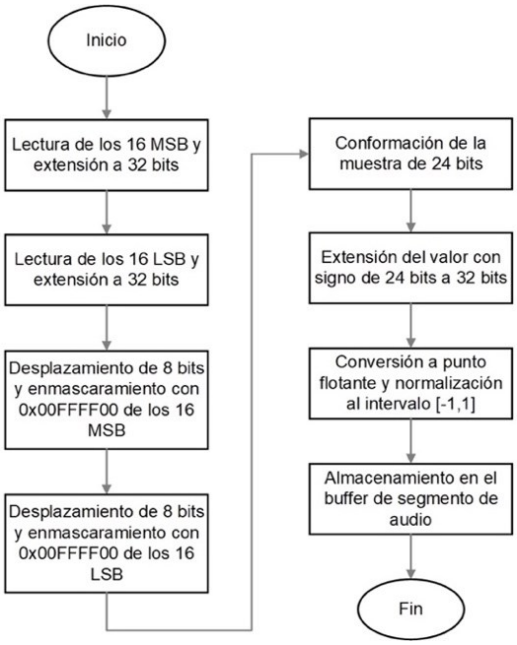


Figura 6

Subrutina de decodificación de una muestra de audio.

La subrutina comienza con la lectura de los 16 MSB y los 16 LSB contenidos en las posiciones adyacentes del buffer, realizando una extensión de cada uno de ellos a valores de 32 bits. Luego, se realizan operaciones de desplazamiento y enmascaramiento de bits para conformar la muestra original de 24 bits transmitida por el micrófono. Esta estará contenida en un dato de 32 bits. Como las muestras de 24 bits transmitidas por el micrófono se encuentran en complemento a 2, al almacenar el valor original de 24 bits en un dato de 32 bits se debe realizar una extensión del valor con signo. Luego, se realiza la conversión a punto flotante y se normaliza para llevar el valor al intervalo [-1,1].

La Fig. 7 muestra el procesamiento completo de una muestra de audio tomando como ejemplo el valor 0x8EAA33.

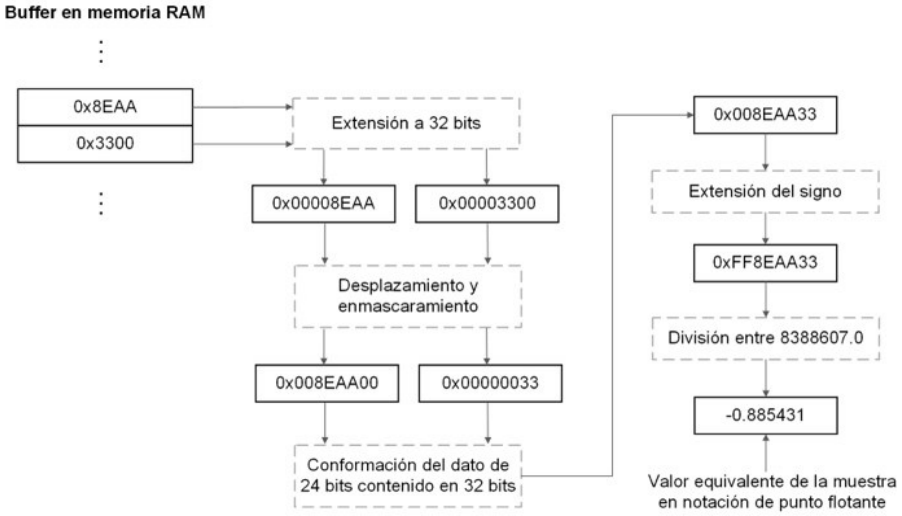


Figura 7

Decodificación de una muestra de audio de valor 0x8EAA33.

Una vez obtenido el valor de cada muestra de audio en notación de punto flotante normalizada, se almacena en el buffer del segmento de audio a procesar.

3.3.- SUBRUTINA DE PROCESAMIENTO DE SEGMENTOS DE AUDIO

La Fig. 8 muestra un diagrama de flujo de la subrutina de procesamiento de un segmento de audio de 30 ms. Esta subrutina se realiza cada vez que se completa el llenado de una nueva mitad del buffer DMA y se terminan de decodificar todas sus muestras.

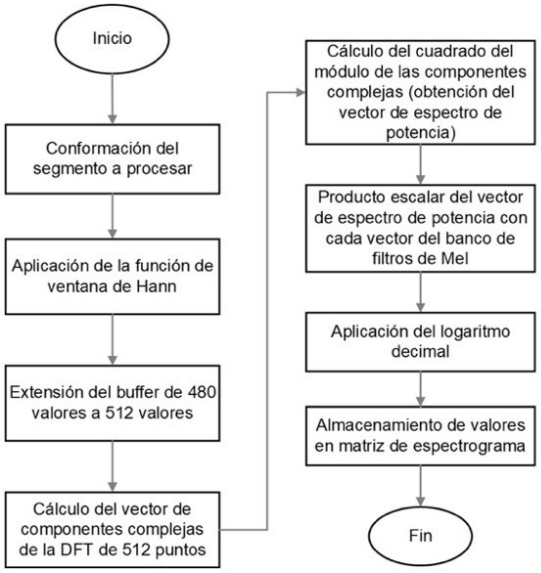


Figura 8

Subrutina de procesamiento de un segmento de audio.

La subrutina comienza con la conformación del segmento de audio a procesar. Cada segmento de audio se conforma con la mitad del buffer DMA llenada anteriormente y la nueva mitad recién llenada. De esta forma, se implementa por *firmware* el solapamiento entre segmentos en cada uno de los segmentos de audio que se conforman.

Una vez conformado el segmento, se aplica el algoritmo de cálculo del espectrograma logarítmico de Mel. Primero, se le aplica al segmento una función de ventana de Hann de 480 puntos. Los valores de la ventana de Hann son almacenados en un vector en memoria de programa por ser valores constantes, para optimizar el uso de la memoria RAM.

El segmento de audio enventanado es almacenado en un buffer de 512 valores rellenando con ceros los valores faltantes. Este buffer es empleado para el cálculo de la DFT de 512 puntos. El algoritmo de la FFT empleado obtiene como resultado un vector donde los valores de las partes real e imaginaria de cada punto de la DFT se almacenan en posiciones consecutivas del buffer. Es por ello que el paso siguiente implica el cálculo del cuadrado del módulo de cada componente compleja almacenada en el vector. El nuevo vector resultante contendrá los valores del espectro de potencia del segmento de audio.

El siguiente paso de la subrutina es realizar la conversión a la escala de frecuencias de Mel a través del banco de filtros de Mel definido. Los valores del banco de filtros son constantes, por lo que también son almacenados en una matriz en memoria de programa con vistas a optimizar el uso de la RAM. La conversión del espectro de potencia a la escala de Mel se realiza calculando el producto escalar de dicho vector por cada vector del banco de filtros. Cada producto escalar corresponde a la energía contenida en cada una de las bandas del espectro de Mel. El número de bandas de energía está determinado por el número de filtros empleados en el banco de filtros.

El número de bandas empleadas en los trabajos mencionados por López-Espejo et al. varía de 20 a 128 [3]. Sin embargo, la experiencia sugiere que el desempeño de los sistemas de detección de palabras clave no es significativamente sensible al valor de este parámetro [21]. Esto permite emplear el menor número posible de canales para reducir las dimensiones del espectrograma de Mel y con ello optimizar el uso de la memoria sin comprometer la precisión del algoritmo clasificador que se implemente posteriormente. De esta forma, se propone el empleo de 20 canales.

Para la implementación de estas operaciones en el microcontrolador se usaron las funciones *arm_mult_f32()*, *arm_rfft_fast_f32()*, *arm_cmplx_mag_squared_f32()* y *arm_dot_prod_f32()* contenidas en el driver CMSIS-DSP de ARM, que implementan rutinas generales de procesamiento digital de señales.

A cada uno de los valores del vector de espectro de Mel obtenido le es aplicado un logaritmo decimal para llevarlos a escala logarítmica. Luego, este vector es almacenado en la posición correspondiente en la matriz de espectrograma de Mel reservada en memoria RAM. De esta forma, se van obteniendo los espectros de Mel en escala logarítmica correspondientes a cada segmento de audio y se va conformando una matriz de espectrograma en tiempo real.

4.- EXPERIMENTACIÓN Y RESULTADOS

Para la validación del sistema se empleó un prototipo experimental del sistema basado en la placa comercial de desarrollo STM32-NUCLEO-G474. Esta placa está basada en el microcontrolador STM32G474RET6, de la serie G4 de micro-controladores de 32 bits de propósito general de STMicroelectronics [22]. Para el uso del micrófono SPH0645LM4H-B se empleó la placa comercial SPH0645 de Adafruit. La Fig. 9 muestra el prototipo experimental empleado.

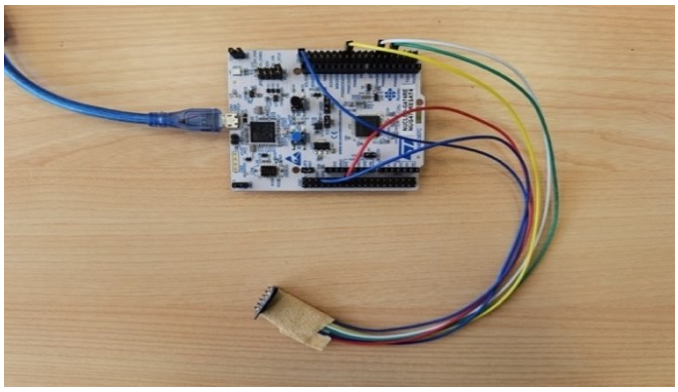


Figura 9

Prototipo usado en la experimentación.

Se diseñó un escenario para la realización de experimentos que permitieran comparar los resultados obtenidos en el cálculo de los espectrogramas en el microcontrolador STM32G474RET6 con los de un software de alto nivel ejecutado en una computadora de altas prestaciones. Para ello, se diseñó un programa en el lenguaje de alto nivel *Python* que implementa la obtención de los espectrogramas de Mel en la computadora de la misma forma en que se realiza en el microcontrolador, con el uso del popular módulo de código abierto *Librosa* [23]. La computadora empleada para la ejecución del software de alto nivel fue una computadora portátil *MacBook Pro M2* basada en el chip *Apple Silicon M2 Pro*, con 16 GB de memoria RAM y un micrófono de un canal MacBook Pro propio de Apple. En la Fig. 10 se ilustra la concepción del experimento realizado.

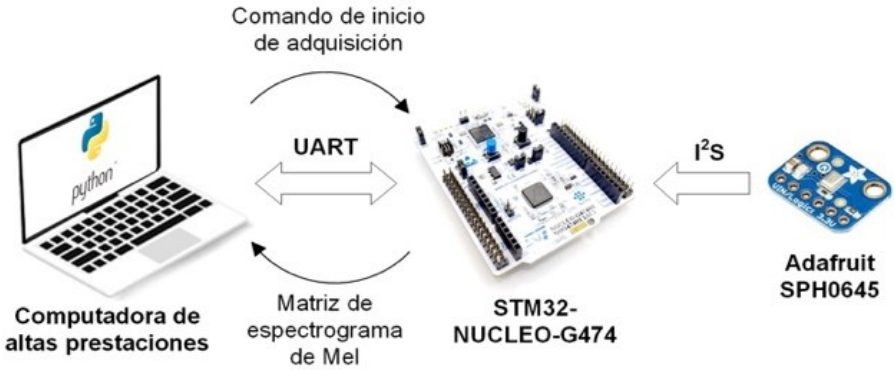
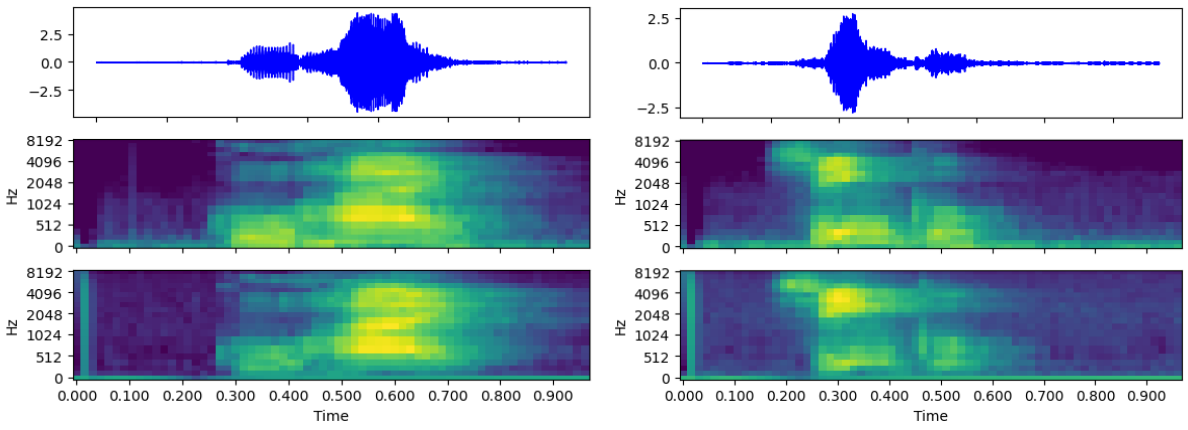


Figura 10

Escenario experimental diseñado.

Para poder comparar los espectrogramas, se partió del principio de lograr adquirir la misma señal de audio en tiempo real en ambas plataformas. Para simultanear la adquisición de audio se estableció una comunicación UART computadora-microcontrolador. Para establecer la comunicación se usó el módulo de código abierto *PySerial* en la computadora y la interfaz LPUART del STM32G474RET6. El software de alto nivel envía un comando de inicio de adquisición por puerto serie desde la computadora al microcontrolador y se comienza a adquirir la señal de audio de forma simultánea en ambos dispositivos. Una vez transcurrido un segundo, se calcula el espectrograma de Mel de la señal adquirida en la computadora y el microcontrolador envía por puerto serie la matriz de espectrograma de Mel computada.

La Fig. 11. muestra los resultados obtenidos para dos señales de audio de 1 segundo de duración que contienen la pronunciación de dos palabras diferentes en idioma español, “cujae” para la Fig. 11 (a) y “cinco” para la Fig. 11 (b). En cada figura se muestra de arriba hacia abajo la forma de onda de la señal adquirida en el tiempo, el espectrograma de Mel obtenido por el software de alto nivel en la computadora y el espectrograma de Mel obtenido en el microcontrolador STM32G474RET6.



(a) **(b)**

Figura 11

Resultados obtenidos para dos señales de audio diferentes. (a) “cujae” (b) “cinco”.

Como se observa en las imágenes, los espectrogramas obtenidos en ambas plataformas son similares. Las ligeras diferencias apreciables pueden estar dadas por las diferencias en las curvas de sensibilidad de los micrófonos y la posición del hablante respecto a ellos y al entorno, lo cual puede introducir determinados armónicos por reflexiones en las señales.

Para comprobar que la simultaneidad en el almacenamiento y procesamiento de los datos en el microcontrolador ocurre en tiempo real sin pérdida de muestras de señal se realizaron varios experimentos en los cuales se midió el tiempo de procesamiento de las muestras de una mitad del buffer ante la llegada de la interrupción correspondiente del DMA. Este proceso se ejecuta en paralelo con el llenado de la otra mitad del buffer, y debe ejecutarse en un tiempo inferior para que no ocurra pérdida de muestras. Para la realización de los experimentos, se empleó en el prototipo descrito uno de los temporizadores de propósito general del microcontrolador STM32G474RET6. En las mediciones realizadas se obtuvo que el tiempo de ejecución de la subrutina de procesamiento es de aproximadamente 805 µs para una frecuencia de operación del microcontrolador de 170 MHz, lo cual valida la solución propuesta al ser este tiempo inferior a la ventana de 15 ms establecida para la duración de los segmentos de audio que se almacenan en cada mitad del buffer.

5.- CONCLUSIONES

En este artículo se presentó una implementación del algoritmo para la obtención de espectrogramas logarítmicos de Mel en tiempo real en microcontroladores de la familia STM32. La estrategia de implementación se basa en el uso de la FPU de los procesadores ARM Cortex-M para acelerar los cálculos matemáticos y un controlador de acceso directo a memoria en conjunto con una interfaz de comunicación I2S para realizar la adquisición. El enfoque propuesto reduce el uso de la memoria RAM. Además, permite realizar el procesamiento del audio adquirido de forma simultánea a la adquisición facilitando el ajuste en tiempo real de la extensión temporal de los espectrogramas. La estrategia presentada fue validada en un sistema experimental basado en el microcontrolador STM32G474RET6.

Desde el punto de vista de la reducción en el uso de la memoria, el empleo de micrófonos con salida digital I2S como el usado en la validación de la propuesta constituye una limitación, puesto que el protocolo I2S está diseñado para transmitir de forma consecutiva y alterna las muestras correspondientes a dos canales de audio. El empleo del controlador DMA para recibir y almacenar las muestras en memoria en forma de bloques obliga a recibir los datos del canal no utilizado y descartarlos por software a la hora de realizar el procesamiento, con lo cual se hace necesario contar con un buffer en memoria del doble de capacidad que el que se necesitaría si se recibieran solamente las muestras del canal útil. Esta limitación se tendría también con el empleo de micrófonos de salida PDM, que constituye la alternativa al I2S más común en este tipo de aplicaciones. No obstante, es posible el empleo del conversor analógico-digital de los STM32 para adquirir las señales a través de micrófonos de salida analógica como electretos, variante que es compatible con el empleo del DMA y, por ende, con la solución propuesta para el cálculo de los espectrogramas haciendo un menor uso de la memoria. Sin embargo, el empleo de este tipo de micrófonos implica lidiar con todas las problemáticas asociadas al canal analógico de adquisición que constituyen el motivo fundamental de la preferencia de soluciones MEMS integradas. Además, los conversores analógico-digitales de aproximaciones sucesivas presentes en los STM32 (con resoluciones del orden de los 12 bits) no alcanzan las resoluciones del orden de 24 bits de los conversores ΔΣ que ofrecen los micrófonos MEMS, con lo cual se degradaría la resolución de las muestras de señal adquiridas. Teniendo estos elementos en cuenta, las subrutinas desarrolladas son compatibles con el uso de cualquier micrófono o códec de audio de salida I2S, pero pueden ser fácilmente adaptadas para emplear otros tipos de interfaces de salida como PDM, que impongan el almacenamiento en memoria de dos canales.

Como proyección futura se propone diseñar e implementar un algoritmo de aprendizaje profundo en el microcontrolador para el reconocimiento de patrones en tiempo real en los espectrogramas, que permita la detección de palabras clave en señales continuas de audio. Se considerará evaluar el impacto que pudieran tener en la precisión del algoritmo el empleo de técnicas de cuantización como las propuestas por Fariselli et al. en [24] para el cálculo de los MFCC o el empleo de Transformadas de Wavelet para la eliminación de ruido en las señales de voz.

AGRADECIMIENTOS

Los autores desean agradecer al Programa de Intercambio y Movilidad Académica (PIMA) auspiciado por la Universidad de Cádiz por hacer posible la colaboración para el desarrollo de esta investigación. Esta investigación se benefició del financiamiento del proyecto de investigación FEDER: “Sistemas multimodales avanzados para prótesis robóticas de miembro superior (PRO-BOTHAND)” (FEDER-UCA18-108407) de la Junta de Andalucía, en España.

REFERENCIAS

1. Liu H, Abhyankar A, Mishchenko Y, Sénéchal T, Fu G, Kulis B, et al. Metadata-Aware End-to-End Keyword Spotting. Interspeech 2020. Shanghai; China; Oct 25-29, 2020. p. 2282–6.
2. Saha SS, Sandha SS, Srivastava M. Machine Learning for Microcontroller-Class Hardware: A Review. IEEE Sens J. 2022 Nov;22(22):21362–90.
3. López-Espejo I, Tan ZH, Hansen JHL, Jensen J. Deep Spoken Keyword Spotting: An Overview. IEEE Access. 2022;10:4169–99.
4. Alías F, Socoró J, Sevillano X. A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds. Appl Sci. 2016 May;6(5):143.
5. Malik M, Malik MK, Mehmood K, Makhdoom I. Automatic speech recognition: a survey. Multimed Tools Appl. 2021 Mar;80(6):9411–57.
6. Prabakaran D, Shyamala R. A Review on Performance of Voice Feature Extraction Techniques. 3rd International Conference on Computing and Communications Technologies (ICCCT). Chennai; India; 21-22 Feb, 2019. p. 221–31.
7. Labied M, Belangour A. Automatic Speech Recognition Features Extraction Techniques: A Multi-criteria Comparison. Int J Adv Comput Sci Appl. 2021;12(8).
8. Ye L, Li Y, Dong W, Seppänen T, Alasaarela E. MCU-based isolated appealing words detecting method with AI techniques. Lect Notes Inst Comput Sci Soc-Inform Telecommun Eng LNICST. 2019; 287:300–8.
9. Miah MN, Wang G. Keyword Spotting with Deep Neural Network on Edge Devices. In: 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC). Beijing; China; 15-17 Jul, 2022. p. 98–102.
10. Rusci M, Tuytelaars T. On-device Customization of Tiny Deep Learning Models for Keyword Spotting with Few Examples. IEEE Micro. 2023;1–7.
11. Li Y, Chang R, Li N. A Wearable Breath Sound Monitoring Device Using the DWT-based MFCC Optimization Algorithm. In: 2023 6th International Conference on Electronics Technology (ICET). Chengdu; China; 12-15 May, 2023. p. 1027–32.

12. Purwins H, Li B, Virtanen T, Schlüter J, Chang SY, Sainath T. Deep Learning for Audio Signal Processing. IEEE J Sel Top Signal Process. 2019 May;13(2):206–19.
13. Dimbiniaina M, Pau DP, Naramo TA. Mel Power Spectrogram Approximation By Tiny Neural Networks for Home Appliances Classification. 2023 IEEE International Workshop on Metrology for Industry 40 & IoT (MetroInd40&IoT). Brescia; Italy; 6-8 Jun, 2023. p. 60–5.
14. Radford A, Kim JW, Xu T, Brockman G, McLeavey C, Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. 40th International Conference on Machine Learning (ICML 2023); Honolulu; Hawaii; USA; Jul 23-29, 2023. p. 28492-28518.
15. Lorenser T. The DSP capabilities of ARM® Cortex®-M4 and Cortex-M7 Processors. ARM; 2016 Nov.
16. Giménez NL, Freitag F, Lee J, Vandierendonck H. Comparison of Two Microcontroller Boards for On-Device Model Training in a Keyword Spotting Task. 11th Mediterranean Conference on Embedded Computing (MECO). Budva; Montenegro; 7-10 Jun, 2022. p. 1–4.
17. CMSIS-DSP software library. ARM; Available from: https://arm-software.github.io/CMSIS_5/DSP/html/index.html
18. Fulop SA. Speech Spectrum Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. (Signals and Communication Technology).
19. Abdul ZKh, Al-Talabani AK. Mel Frequency Cepstral Coefficient and its Applications: A Review. IEEE Access. 2022;10:122136–58.
20. Knowles. Datasheet SPH0645LM4H-B Rev C. 2017.
21. López-Espejo I, Tan ZH, Jensen J. Exploring Filterbank Learning for Keyword Spotting. In: 2020 28th European Signal Processing Conference (EUSIPCO). Amsterdam; Netherlands; 18-21 Jan, 2021. p. 331–5.
22. STMicroelectronics. Datasheet STM32G474xB STM32G474xC STM32G474xE. 2021 Nov. DS12288 Rev 6.
23. McFee B, Raffel C, Liang D, Ellis D, McVicar M, Battenberg E, et al. librosa: Audio and Music Signal Analysis in Python. 14th Python in Science Conference. Austin; Texas; Jul 6-12, 2015. p. 18–24.
24. Fariselli M, Rusci M, Cambonie J, Flamand E. Integer-Only Approximated MFCC for Ultra-Low Power Audio NN Processing on Multi-Core MCUs. In: 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS). Washington DC; USA; 6-9 Jun, 2021. p. 1–4.

CONFLICTO DE INTERESES

Ninguno de los autores manifestó la existencia de posibles conflictos de intereses que debieran ser declarados en relación con este artículo.

CONTRIBUCIONES DE LOS AUTORES

Alejandro Perdomo-Campos: contribución importante en la conceptualización de la solución y en el diseño de la implementación descrita del algoritmo. así como en el proceso de investigación: la implementación del diseño conceptualizado, el desarrollo del software embebido y la realización de las pruebas para la comprobación de los resultados. Desempeñó un papel fundamental en la redacción del artículo y en la discusión y presentación de los resultados.

Jorge Ramírez-Beltrán: contribución importante en la metodología y la supervisión de la investigación. Desarrolló un papel fundamental en la validación-verificación de los resultados obtenidos y en la revisión del artículo.

Arturo Morgado-Estevez: contribución importante en la supervisión de la investigación, la adquisición de fondos para su desarrollo y la administración del proyecto. Desarrolló un papel fundamental en la validación-verificación de los resultados obtenidos y en la revisión del artículo.

AUTORES

Alejandro Perdomo-Campos, Ingeniero en Telecomunicaciones y Electrónica. Profesor en adiestramiento del Centro de Investigaciones en Microelectrónica. Universidad Tecnológica de La Habana "José Antonio Echeverría" (Cujae), La Habana, Cuba, aperdomoc@tele.cujae.edu.cu, No. ORCID: 0000-0002-6253-875X. Miembro del Grupo de Instrumentación Electrónica y Sensores Inteligentes del Centro de Investigaciones Hidráulicas, Cujae. Investiga en instrumentación, procesamiento digital de señales, reconocimiento de patrones y sistemas empotrados.

Jorge Ramírez-Beltrán, Ingeniero Electrónico. Doctor en Ciencias Técnicas. Investigador Titular. Universidad Tecnológica de La Habana "José Antonio Echeverría" (Cujae), La Habana, Cuba, jorgeramirezcihcujae@gmail.com, No. ORCID: 0000-

0002-4125-2656. Jefe del Grupo de Instrumentación Electrónica y Sensores Inteligentes del Centro de Investigaciones Hidráulicas, Cujae. Investiga en instrumentación y detección de eventos en tuberías de agua.

Arturo Morgado-Estevez, Doctor Ingeniero Industrial. Escuela Superior de Ingeniería de la Universidad de Cádiz (UCA), Andalucía, España, arturo.morgado@uca.es, No. ORCID: 0000-0002-3639-3649. Responsable del Grupo de Investigación de Robótica Aplicada de la Escuela Superior de Ingeniería de la UCA. Investiga en sistemas robóticos bioinspirados en el diseño y desarrollo hardware/software y en robótica industrial y educativa.