UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:

INGENIEROS ELECTRÓNICOS

TEMA:

DESARROLLO DE UN ALGORITMO DE RECONOCIMIENTO DE PALABRAS MEDIANTE UN MICROCONTROLADOR PARA GIECA

AUTORES:

JHON CARLOS CUSCO SÁNCHEZ

JOHN EFRAÍN LEÓN ALBÁN

TUTOR:

LUIS GERMÁN OÑATE CADENA

Quito, agosto de 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Jhon Carlos Cusco Sánchez con documento de identificación N° 1716468002 y John Efraín León Albán con documento de identificación N° 0502935935, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: "Desarrollo de un algoritmo de reconocimiento de palabras mediante un microcontrolador para GIECA", mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado por la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Jhon Carlos Cusco Sánchez

Cédula: 1716468002

John Efraín León Albán

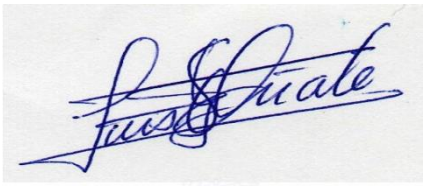
Cédula: 0502935935

Fecha: Quito, agosto de 2020.

DECLARATORIA DE COAUTORIA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, "Desarrollo de un algoritmo de reconocimiento de palabras mediante un microcontrolador para GIECA", realizado por Jhon Carlos Cusco Sánchez y John Efraín León Albán, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, agosto de 2020.



_____

Luis Germán Oñate Cadena

Cédula de identidad: 1712157401

DEDICATORIA

Mi tesis la dedico con todo mi amor y cariño a toda mi familia, por el sacrificio y esfuerzo que cada uno ha tenido que pasar, para poder llegar a la culminación de mi carrera profesional, aunque hemos pasado por épocas difíciles nunca me dieron la espalda y me supieron apoyar de una u otra manera para seguir adelante siempre con amor y comprensión, mis padres personas nobles y sencillas, los cuales me ayudaron a que todo esto se pueda cumplir de la mejor manera y por su incondicional apoyo en todo momento, a mi esposa e hijo que son mi principal fuente de motivación, que fueron los que me impulsaban día a día a superarme para que la vida nos depare un futuro mejor . También quiero dedicar este trabajo a la familia **SALESIANA,** para todos ellos tengo una enorme gratitud en el corazón y doy las gracias a Dios por haber podido conocerlos y demostrar que los objetivos por más difíciles que se nos presentaban los podemos superar con esfuerzo y dedicación.

Jhon Carlos Cusco Sánchez

Este proyecto está dedicado para toda mi familia y como una muestra de cariño que se puede tener hacia mis padres los cuales me ayudaron que todo esto se pueda cumplir de la mejor manera y por su incondicional apoyo en todo momento, me encuentro culminando mi carrera. También quiero dedicar este trabajo a la familia **SALESIANA,** para todos ellos tengo una enorme gratitud en el corazón que me impulsa a seguir adelante, también dedico este trabajo a todos los que me conocen como una muestra que con perseverancia se alcanza los objetivos y las cosas buenas no son fáciles y no llegan con brevedad.

John Efraín León Albán

AGRADECIMIENTO

Agradezco primero a Dios por permitirme llegar a este día y a mis padres que mediante su sacrificio tanto económico como espiritual han sabido encaminarme a lo largo de mi vida.

Un sincero agradecimiento a mi tutor, el Ing. Luis Germán Oñate Cadena por la colaboración dada en el desarrollo de este proyecto, a mis compañeros de la universidad con quienes fue más agradable adquirir, recibir y compartir el conocimiento en las aulas.

Por último y no menos importante a todos los miembros de mi familia que nunca dejaron que me retire de esta meta cumplida.

Jhon Carlos Cusco Sánchez

Ante todo, me gustaría expresar mi sincero agradecimiento a mi tutor, el Ing. Luis Germán Oñate Cadena por el continuo apoyo de mi estudio e investigación en el proyecto de tesis, por su paciencia, motivación, entusiasmo y gran conocimiento. Su orientación me ayudó en todo el tiempo de investigación y creación de este proyecto.

Por último y más importante, me gustaría agradecer a mis padres por darme a luz en primer lugar y apoyarme espiritualmente a lo largo de mi vida.

John Efraín León Albán

INDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el proyecto Desarrollo de un algoritmo de reconocimiento de palabras mediante un microcontrolador para GIECA, se estudia un prototipo que es capaz de reconocer palabras específicas mediante un micrófono, un microcontrolador y emite una respuesta sonora acorde a la palabra dictada.

Para conseguir el objetivo antes citado se implementará un prototipo electrónico para el reconocimiento de voz utilizando un microcontrolador KPU-K210 como procesador central del circuito electrónico que tiene dos métodos de programación, el primero grabará las repuestas que serán reproducidas en el altavoz del prototipo, almacenando estos audios en una tarjeta SD de 64 gigas, el segundo grabará los comandos de voz que van a ser comparados y extraerá un vector cepstral con características propias de cada uno , para compararlo con los comandos de voz que el hablante desee reconocer al igual que a los comandos antes almacenados en la memoria extraerá características propias de cada uno y determinará un vector cepstral para compararlo . Para sensar palabras habladas se utilizó un módulo de audio con alta sensibilidad I2S-APU el cual se lo conecta a una entrada del prototipo Maix Go. Una vez que se procesa la señal de audio se emite una respuesta audible a través del módulo reproductor IMAFDC RISC-V, enviando una señal desde el microcontrolador por comunicación serial. De la misma manera que se envía una señal por el puerto serial al reproductor de audio, se envía una señal a una PC para que presente la respuesta en texto de la palabra reconocida.

ABSTRACT

In this project a prototype is studied that is capable of recognizing specific words that are heard through a microphone, a microcontroller and a sound response according to the dictated word. To achieve the aforementioned objective, an electronic prototype for voice recognition has been implemented using a KPU K210 microcontroller as the central processor of the Maix Go electronic circuit where 2 programming methods are stored, the first one will only record the responses that will be reproduced in the loudspeaker. From the prototype stored these audios on a 64 gigabyte SD card, the second method will record the voice commands that will be compared and will extract a cepstral vector with its own characteristics, to compare it with the voice commands that the speaker recognizes that Like the commands previously stored in memory, it will extract characteristics of each one and determine a cepstral vector to compare it. To sense spoken words, a highly sensitive I2S APU (audio processor) audio module is processed, which will be connected to one input of the Maix Go prototype. Once the audio signal is processed it emits an audible response through the IMAFDC RISC-V player module by sending a signal from the microcontroller via serial communication. In the same way that it sends a signal through the serial port to the audio player, it will send several a signals to a PC to present the text response of the recognized word.

INTRODUCCIÓN:

En los últimos años varios algoritmos de reconocimiento de comandos de audio y voz tienen presencia en el campo de investigación de la ingeniería desde los años 50 hasta la actualidad. Según Oropeza (2006) un resumen de la historia de la investigación en este campo se presenta a continuación en la tabla 2.1.

Para desarrollar un algoritmo de reconocimiento de comandos de audio y voz se lo realizará en dos fases, que son: Entrenamiento y Reconocimiento. El entrenamiento es la etapa que genera un reto ya que no es una tarea sencilla y es la parte que fundamental del funcionamiento del sistema. En el trabajo de Oropeza (2006) se utiliza un Sistema Basado en Conocimiento (SBC) que permite clasificar la señal de entrada en unidades silábicas utilizando las reglas lingüísticas del español ya que depende del contexto y de la prosodia. (Oropeza Rodríguez & Suárez Guerra, 2006)

Según José Oropeza el Sistema de Reconocimiento que funciona Automáticamente con el Habla es el que tiene la capacidad de tratar la señal audible pronunciada por una persona. Para esto, el autor menciona que el reconocimiento de la voz en español se puede hacer si se entiende la teoría de los fonemas de las sílabas. (Oropeza Rodríguez & Suárez Guerra, 2006)

El presente documento está conformado por los siguientes capítulos:

- El primer capítulo presenta los antecedentes que produjeron el trabajo el presente trabajo, junto con su justificación y una solución propuesta.
- El capítulo dos enmarca todos los fundamentos teóricos que fueron necesarios para el desarrollo del presente trabajo.
- El capítulo tres presenta el desarrollo del presente proyecto y detalla las estrategias utilizadas, así como lo métodos y técnicas.
- El capítulo cuatro expone las pruebas, resultados, conclusiones y recomendaciones que se obtuvieron con el presente trabajo.

CAPÍTULO 1
ANTECEDENTES

1.1 PLANTEAMIENTO DEL PROBLEMA

En el área de investigación de la facultad de Ingeniería electrónica de la Politécnica Salesiana se trabaja en el proyecto "Sistema robótico de soporte para personas discapacitadas visualmente utilizando técnicas de visión artificial y control automático" por parte del grupo GIECA, el cual busca guiar a una persona discapacitada de la vista con varios módulos acoplados que realizarán mediciones de distancias directas hacia objetos específicos que se encuentren en su entorno para evitar colisiones con objetos delicados o la búsqueda de cosas específicas con características propias en el medio , al llevar a cabo esta búsqueda se necesita tener una comunicación hablada del usuario con el proyecto para lo cual existen varios métodos de comunicación como por ejemplo Google Search Voice o Siri que son plataformas de reconocimiento de patrones de voz pero dependen de una conexión a internet todo el tiempo de su funcionamiento que ocasionarían gastos económicos elevados en el sistema. Existen también sistemas embebidos que reconocen patrones de voz desarrollados en Matlab y Raspberry PI que dependen de sistemas operativos y licencias propias para su funcionamiento, estos sistemas producen un incremento de costos en la fabricación del modelo y tamaño del prototipo.

1.2 JUSTIFICACIÓN DEL PROYECTO

En el proyecto "Sistema robótico de soporte para personas discapacitadas visualmente utilizando técnicas de visión artificial y control automático" existen varios prototipos como lo son: el guiado de personas por cámaras y la ayuda por visión artificial y el complemento más eficaz de ayuda al usuario no vidente, sería tener la facilidad de dar indicaciones habladas al prototipo final para poder ubicar objetos, lugares, etc., que complementarían el funcionamiento de los módulos mencionados y así lograr que el prototipo sea muchísimo más eficaz al momento de brindar asistencia al usuario no vidente, para que pueda desarrollar una mayor independencia personal, disminuyendo así la necesidad de tener siempre un acompañante que lo guie día a día en su diario vivir, con la culminación total de este prototipo se lograra que el usuario

no vidente sea capaz de incorporarse a una actividad laboral con una mayor seguridad en sí mismo para poder cumplir a cabalidad sus obligaciones.

1.3 OBJETIVOS

1.3.1 Objetivo general:

Desarrollar un algoritmo para el reconocimiento de patrones de voz de una persona mediante un microcontrolador, proporcionando una respuesta tipo texto plano enviada a través comunicación serial y convertida a audio por un sintetizador.

1.3.2 Objetivos específicos:

- Analizar el estado del arte de sistemas de reconocimiento de voz para la identificación de las variables que intervienen en el proceso.
- Desarrollar un algoritmo para el reconocimiento de comandos de voz mediante un lenguaje de programación para microcontroladores.
- Implementar un prototipo electrónico utilizando un microcontrolador para la interacción con una persona.
- Realizar pruebas con el prototipo de reconocimiento de comandos de voz implementado para la validación de su funcionamiento con una persona.

1.4 PROPUESTA DE SOLUCIÓN

Se desarrollará el algoritmo para reconocimiento de comandos de voz para reconocer palabras específicas como, por ejemplo: puerta, silla, mesa, pelota, entre otros, la cantidad de palabras a identificar serán definidas por los requerimientos del proyecto desarrollado por GIECA.

Para evitar el uso de Sistemas Operativos propios o conexión a internet el algoritmo de reconocimiento de patrones de voz será programado en un microcontrolador y enviará una respuesta en texto plano a través de comunicación serial al sistema general del proyecto, además proporcionará una respuesta audible pre programada en función de cada palabra mediante un altavoz.

CAPÍTULO 2
MARCO TEÓRICO

2.1 ESTADO DEL ARTE

Se han desarrollado en los últimos años varios trabajos de investigación en este campo, de los cuales se presenta un resumen a continuación.

(Martínez, 2013), Presentan en su trabajo Titulado "Reconocimiento de comandos voz con técnicas de Coeficientes Cepstrales de Mel, SBC y de Espectrogramas "Los conflictos que representan al usar estos sistemas de reconocimiento son las variaciones en la voz. Usualmente, los seres humanos tienen cambios consientes o inconscientes que pueden ocasionar errores en el método, además de esto las variaciones de voz pueden ser generadas de forma natural y artificial. El documento presenta una técnica de reconocimiento en paralelo, manejando tres técnicas de reconocimiento: Coeficientes Cepstrales de Mel, SBC y de espectrograma. Utilizando un sistema vectorial de tipo archivador, las técnicas mencionadas emiten un conjunto de individuos con las posibilidades acertadas y luego de su valoración, se tomará la mejor opción al evaluar las técnicas mencionadas. Como resultado se obtuvo un 93.33% de acierto en el reconocimiento.

(De Luna, 2006), El documento presenta un reconocimiento de voz usando MODELOS DE TIPO OCULTO DESAROLLADO POR MARKOV, RED DE TIPO NEURONAL EN VISION ARTIFICIAL YDE ALINEAMIENTOS DINÁMICOS EN EL TIEMPO, para desarrollar un sistema de reconocimiento. Con el fin de establecer coincidencias de pronunciaciones y ejecutar un sistema de reconocimiento eficaz, sobresalen las ANN, el DTW y Los HMM. Para realizar este proyecto se utilizan los tres algoritmos, debido a que las ANN indican un notable aprendizaje de las señales de entrada, dando adaptación a los cambios que muestra la voz, lo que servirá como ayuda al algoritmo, el DTW porque es un sistema eficiente en varios trabajos y permitan ser tomados en nuestro proyecto, y por último HMM por ser un algoritmo completo y usado por la mayoría. El porcentaje de reconocimiento con estas técnicas son las siguientes al aplicar Redes Neuronales Artificiales se obtuvo un 81%,

con el algoritmo de Alineamiento Dinámico del Tiempo se alcanzó un 95%, y finalmente con el modelo Modelos Ocultos de Markov se consiguió 84%.

(Cruz, 2017), Presenta su trabajo "Reconocimiento de Voz usando Redes Neuronales Artificiales Backpropagation y Coeficientes LPC" donde se plantea la técnica de reconocimiento de individuos en una señal telefónica. La técnica se basa en la conducta de las Red Neuronal Artificial (RNA), en especial, al algoritmo Backpropagation. La validación de individuos en una señal telefónica, usando el modelo de voz, es lo primordial para la realizar este modelo de voz, Coeficientes LPC y Redes Neuronales. Este método busca ser efectuado en casos jurídicos en la que se presenta una grabación desde un teléfono del procesado, que serviría como constancia y se utilizará una técnica que de un diagnóstico de si evidentemente la voz que se halla en la grabación concierne al procesado, al realizarse las pruebas se obtuvo un reconocimiento del 100%.

(González, 2019), Muestra una investigación titulada "Evaluación comparativa de sistemas de reconocimiento de locutor basados en los algoritmos LPC, CC y MFCC" que plantea ejecutar una valoración de técnicas de reconocimiento de personas fundados en el algoritmo de Coeficientes por Predicciones Lineales, Coeficientes de la Cepsis y coeficientes obtenidos por Frecuencias Mel), usados para extraer cuantificaciones de audio. La valoración, consiste en establecer las variaciones de desempeño a partir de una señal de entrada que es mostrada a varios ambientes con ruido, por lo tanto, a diferentes niveles de SNR, se comparan los resultados de ejecución para dos individuos. A pesar de que todas las técnicas reducen su desempeño en lugares con ruido, cada uno tiene una forma exclusiva para cierto nivel. Esta valoración valdrá de referencia para creación de nuevas técnicas para reconocimiento para individuos, los cuales circunscriban métodos de perfeccionamiento de voz para disminuir el eco.

(Tiang, 2007), Muestra su trabajo "Implementación de un reconocedor de voz con un microcontrolador MCS51 para controlar una silla de ruedas" En el que se utiliza un microcontrolador de ATMEL AT89C51RC para que una silla de rueda pueda ser controlada por comandos de voz. Esta técnica cuenta con dos métodos para reconocer la voz, El primero es la Distancia Cuadrática Euclidiana que se utilizó para reconocer la voz mediante un modelo de palabras dadas por el algoritmo Markov y el otro es un

Codificador Predictivo Lineal que es utilizado como método de extracción de características. Se obtuvo un 78.57% de reconocimiento con este proyecto.

(Raczynski, IEEE XPLORE, 2018), presenta su investigación titulada “Algoritmo de procesamiento de voz para el reconocimiento de palabras aisladas”, en el que se pretende reducir los costosos cálculos que debe procesar una computadora para realizar dichas tareas de reconocimiento. Para lo cual, en el trabajo se presenta un algoritmo reconocimiento de voz simple que es capaz de reconocer una palabra hablada de un grupo de palabras presentadas previamente enfocado para ser utilizado por ejemplo en un control de dispositivos por señales de audio. Este algoritmo se basa en analizar varias señales evaluadas sobre el dominio del tiempo, donde obtienen características mediante una aproximación lineal para ser comparadas con patrones almacenados en memoria utilizando MATLAB. Con este método el usuario puede almacenar en memoria las palabras que desea que se reconozcan. El porcentaje de reconocimiento de esta técnica es de 80%.

(Oropeza, 2006) Todavía, existen algunas dificultades ocultas en el uso de fonemas, porque a menudo es difícil encontrar el límite entre ellos en la representación acústica del habla. Este trabajo ha proporcionado una alternativa al reconocimiento de voz durante algún tiempo y analiza cómo los ejemplos de sílabas responden a dicho trabajo en español. Durante el experimento, se verificaron tres elementos básicos de la tarea de fraccionamiento: Funciones con Energías de Tiempo Corto, Funciones con Energía de alta frecuencia Cepstrales, y Sistemas que están basados en el conocimiento. El sistema que está basado por los conocimientos y la energía total en poco plazo se usan en un sistema digital, y los resultados obtenidos al usar únicamente el método de energía total a poco plazo es 90,58 %. Al usar el método de energía total de poco tiempo del parámetro y las energías RO de la medida, la tasa de reconocimiento es 94.70%. Esto resultó en un aumento del 5% en comparación con el uso de palabras completas en un corpus sensible al argumento. En cambio, cuando se usa el corpus continuo de laboratorio de habla cuando se usa el método de energías totales y la función basada en el conocimiento de corto tiempo, las tasas de reconocimiento cuando se usan estos tres idiomas son 78.5% y 80.5%, respectivamente. El modelo de lenguaje utilizado en este caso es un grupo de dos letras, en el cual se utiliza Markov

oculto con densidad fija con tres y cinco estados, y cada estado tiene 3 mezclas gaussianas.

Basándose en los artículos citados hemos elaborado una tabla con los distintos métodos de reconocimiento y su efectividad como se puede verificar en la siguiente tabla 2.1.

Tabla 2.1. Sistemas de reconocimiento de voz y su efectividad.

Método de reconocimiento	**Efectividad**
MFCC, SBC Y ESPECTOGRAMAS	93,33%
Red Neuronal Artificial Alineamiento Dinámico del Tiempo Modelos Ocultos de Markov	81% 95% 84%
Reconocimiento de audio por Redes Neuronales y visión Artificial	100%
Microcontrolador MCS51	78,57
Obtención de características mediante MATLAB	80%

Se representa los porcentajes de efectividad de varios sistemas de reconocimiento Fuente: Cusco, J., & León , J. (s.f.).

2.2 TÉCNICA PARA RECONOCIMIENTO DE COMANDOS DE VOZ

Al desarrollar una técnica de reconocimiento de voz se tendrán en cuenta dos fases en este proceso que son: Entrenamiento y Reconocimiento. El entrenamiento es la etapa que genera un reto ya que no es una tarea sencilla y es la parte de la cual depende directamente el éxito de funcionamiento del sistema. En el trabajo de Oropeza (2006) se utiliza un Sistema Basado en Conocimiento (SBC) que permite clasificar la señal de entrada en unidades silábicas utilizando las reglas lingüísticas del español ya que depende del contexto y de la prosodia. (Oropeza Rodríguez & Suárez Guerra, 2006).

Según Oropeza (2006) existen muchos métodos para entrenar un sistema como por ejemplo los que se mencionan a continuación:

- Banco de filtro

- Codificaciones Predictivas Lineales
- Modelo Oculto de Markov
- Rede Neuronal Artificial
- Extracción de características por MEL
- Sistema de reconocimiento Híbrido, entre otros.

Oropeza (2006) hace una observación muy importante en cuanto a la frecuencia con la que se emite una señal de audio y la clasifica en rangos de 8 y 16 KHz. Una vez que la señal se ha digitalizado es importante extraer la señal característica relevantes para tratarla. Para extraer esas características se pueden utilizar por ejemplo las técnicas expuestas a continuación. (Oropeza Rodríguez & Suárez Guerra, 2006).

- Técnica de Fourier
- Codificaciones Predictivas Lineales
- Análisis del coeficiente Cepstral
- Predicción lineal Perceptiva

Un criterio importante al momento de capturar los datos es considerar la frecuencia de muestreo, ya que esta característica es a menudo uno de los principales problemas para reconocer palabras. Pero, aunque se tuviera en cuenta lo dicho antes aún quedan muchos más retos por resolver en este proceso, como por ejemplo los mencionados a continuación. (Oropeza Rodríguez & Suárez Guerra, 2006).

- Tamaños y confusiones del vocabulario.
- Sistema independiente y dependiente del locutor.
- Voces aisladas de procedencia discontinua y continua.
- Voces aplicadas en tareas específicas y generales.
- Audio de lectura directa y espontánea.
- Condiciones varias no influyentes.

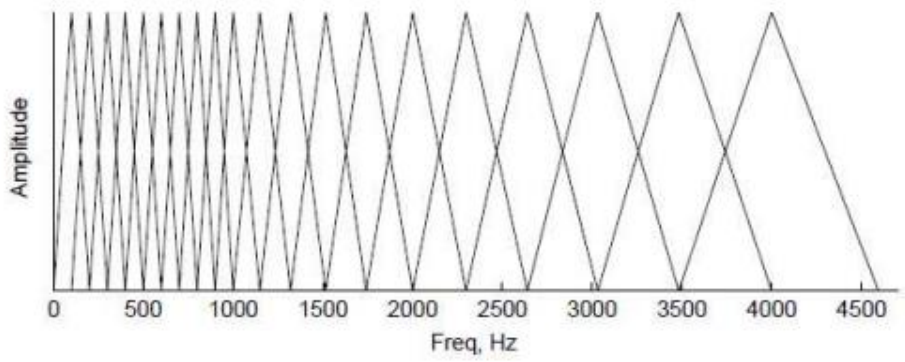
2.3 COEFICIENTES CEPSTRALES Y CARACTERÍSTICAS DE LAS FRECUENCIAS DE MEL)

Imitando lo que ocurre en el medio auditivo humano, el reconocimiento de sonido se realiza en los dominios de la frecuencia. De varias técnicas de parametrización del habla, la más utilizada es el coeficiente de cepstrum de frecuencia Mel o MFCC. Esta

tecnología de segmentación del habla es la más aplicada en los métodos automáticos de reconocimiento de audio.

(Mermelstein & S, Agosto 1980) Implantaron el término "Mel Frecuency Cepstral Coefficient "En los años 80, combinaron el filtro triangular de la distribución de percepción y transformación discreta de coseno y el logaritmo para la energía en la salida del filtro. D&M amplió este trabajo en publicaciones ordinarias para reforzar su uso. No obstante, D&M solo proporciona una visión general del algoritmo (la señal se convierte en el dominio de la frecuencia mediante DFT), además escalar un espectro a través de un conjunto de filtro triangular que está distribuido entre el eje de la frecuencia lineal y logarítmica. Luego, usando la Transformación discreta de coseno (DCT) para comprimir logarítmicamente y transformar la energía de salida de cada filtro para adquirir algún coeficiente de tipo cepstral. Suministraron una imagen del ejemplo de los bancos de filtros como se observar en la (figura 2.1) junto con la siguiente ecuación *Ec. (0.1).*

Figura 2. 1Banco de Filtros



Banco de filtros utilizado, 10 están linealmente espaciados entre 100 y 1000 Hz. Fuente: (Mermelstein & S, Agosto 1980).

En el ancho de banda de un filtro triangular en MFCC se determinan las reparticiones de las frecuencias centrales en cada uno de los filtros, que son funciones de las frecuencias del muestreo y la cantidad de filtros. Además, aumentando la cantidad de filtros en el banco, disminuye el ancho de banda en cada uno de los filtros. No obstante, las particularidades del banco de filtros de los coeficientes de MEL provienen del audio humano, además D&M no explica el número ni la forma de cada filtro, la elección de los factores de superposición por cada filtro adyacente, y no determina

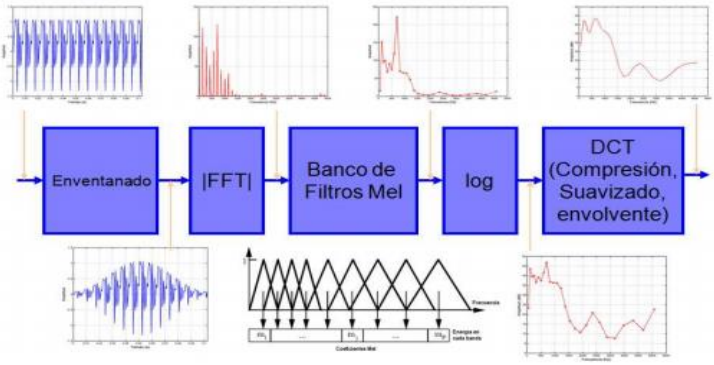
cómo adaptarse al diseño principal. Experimentación con comandos de audio a frecuencias muestreadas distintas a las de 10 kHz. Su característica forma triangular filtrada en por MEL se acerca al modelo de las bandas de pasos naturales de la banda de frecuencia crítica en el oído humano, sin embargo, la relación entre las frecuencias centrales y el ancho de bandas críticas no son utilizadas para determinar el ancho del ruido. La frecuencia del filtro puede expresarse mediante la fórmula (2.1.).

$$MFCC_i = \sum_{K=1}^{20} x_K \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{20}\right] i = 1{,}2, \ldots, M \qquad \textit{Ec. (2.1)}$$

Para lo cual M simboliza la cantidad del coeficiente cepstral y en donde Xk representará los logaritmos de las energías y la salida de los filtros k-ésimos. Tal como se muestra en la Figura 2.7, el punto final de cada filtro está definido por la frecuencia central del filtro adyacente. El banco de memoria consta de veinte filtros, 10 están linealmente espaciados entre 100 y 1000 Hz, y el espacio logarítmico es 5 desde 1 kHz a 2 kHz, los otros cinco son intervalos logarítmicos desde 2 kHz a 4 kHz.

El coeficiente MFCC representa la envoltura de la señal de voz en forma de espectro, logrando así el reconocimiento de voz, el coeficiente C0 muestra las energías de las señales que se pueden usar según su aplicación. Además, el coeficiente C1 posee la explicación moderada y puede usarse como un guía del balance energético completo entre baja y alta frecuencia. Para conseguir más información, la coarticulación fonética, debe ingresar datos sobre la rapidez y velocidad de cada parámetro. Esto produce MEL Delta y MEL Delta-Delta que constituyen las evoluciones temporales del fonema con transición hacia diferentes fonemas. Delta MEL se calcula por el cambio en el coeficiente MFCC en relación con el instante. Por lo tanto, se denominan coeficientes de velocidad (porque cambian con el tiempo) o primeras derivadas. Los coeficientes ΔΔMFCC muestran el cambio en los coeficientes de velocidad, razón por la cual se denominan coeficientes de aceleración y se observan en el siguiente diagrama de bloques de la (Figura 2.2). (Rojo, 2011).

Figura 2. 2 Diagrama de coeficientes de MEL MFCC



Proceso de extracción de coeficientes de MEL Fuente: (Rojo, 2011).

El parámetro del coeficiente MEL es el tipo específico del coeficiente de tiempo cepstrum en la ventana de señal de voz. Para analizar el cepstrum del modelo matemático, se puede decir que es un operador matemático que convierte la convolución de los tiempos en la suma de los dominios espectrales para separar los componentes informativos de las señales del habla, la perturbación y el canal, y luego invertiremos El espectro está determinado con la transformada inversa de Fourier de un logaritmo que está incluido en las señales de audio expresadas en la ecuación (2.2).

$$Cepstrum\ (s[n]) = \hat{s}[n] = F^{-1}[log(|F[s[n]\ ]\ |)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} log\left(\left|Se^{jw}\right|\right) dw \qquad \text{Ec. (2.2)}$$

Entre ellos, s [n] expresa las convoluciones de la excitación y el canal de sonido, expresada en la fórmula (2.3).

$$\mathrm{s[n]} = e[n] * h(n) \qquad \text{Ec. (2.3)}$$

Como se indicó anteriormente, se puede demostrar que después de que se aplica el cepstrum, la convolución se convierte en la suma del dominio del cepstrum (por la cual, el cepstrum se considera una transformación homomórfica), que se puede observar en la fórmula (2.4).

$$\hat{s}[n] = \hat{e}[n] + \hat{h}[n] \qquad \text{Ec. (2.4)}$$

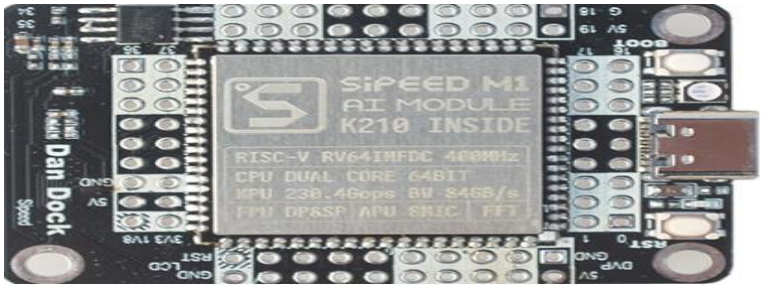
No obstante, si bien todos los sistemas de extracción y parametrización de características de señal de voz usan cepstrum, rara vez se usa cepstrum de forma directa debido a su vulnerabilidad elevada bajo la influencia del conducto, y es muy útil

además de mejorar la eficiencia del sistema. Intenta simular el comportamiento de la frecuencia del oído humano. Por lo tanto, surgió el criterio de MEL, que utiliza unas nuevas escalas de frecuencias no lineales MFCC que copiará el comportamiento del oído humano bajo tonos puros de diferentes frecuencias. Varios estudios en este campo científico han confirmado que el método auditivo humano puede procesar señales del habla en el dominio espectral porque tiene una resolución más alta a bajas frecuencias, que es lo que MEL puede lograr, y tiene una mayor correlación con las frecuencias bajas. Al igual que el sistema auditivo humano. (Mermelstein & S, Agosto 1980).

2.4 TARJETA MAIX GO DE SIPEED

El prototipo de desarrollo de MAIX GO de Sipeed tiene un chip procesador de doble núcleo CPU RISC-V de doble núcleo independiente, que trabaja con un sistema de 64 bits con una Unidad de tipo flotante. Que es conocida por su coprocesador de tipo matemático, este elemento es la unidad principal de procesamientos se dedica a calcular operaciones de punto flotante. Las operaciones básicas que pueden realizar todas las FPU son la suma y la multiplicación cálculo trigonométrico y exponencial. A la que se puede integrar visión artificial y matrices de micrófonos. El procesador K210 es el microcontrolador central de este prototipo que trabaja con código de programación abierto y su programación puedes ser manejada a través de Micro Python que facilita la programación en hardware y código de programación de Arduino. (SIPEED, 2018) microcontrolador que lo alcanzamos a visualizar en la figura (2.3).

Figura 2. 3 Microcontrolador KPU K210



Microcontrolador fabricado por SIPEED de 64 bits con filtrado inteligente, FPU FFT incorporado.

Existen varios modelos que son compatible con MAIX PY que se adaptan a las necesidades del desarrollo de varios prototipos, para cumplir con los objetivos

planteados basados en adquisición de datos de audio, se ha adquirido complementos del prototipo para ensamblarlos y crear un prototipo capaz de procesar varios datos de transmisión y recepción de comandos de voz a continuación se presenta un listado de las partes del prototipo que esta ensamblado y será capaz de compilar y ejecutar sin ningún problema nuestro algoritmo de programación y así poder desarrollar más algoritmos de aplicaciones para el usuario. Complementos que se los observa en la Tabla (2.2) (SIPEED, 2018).

Tabla 2.2 Complementos del prototipo MAIX GO

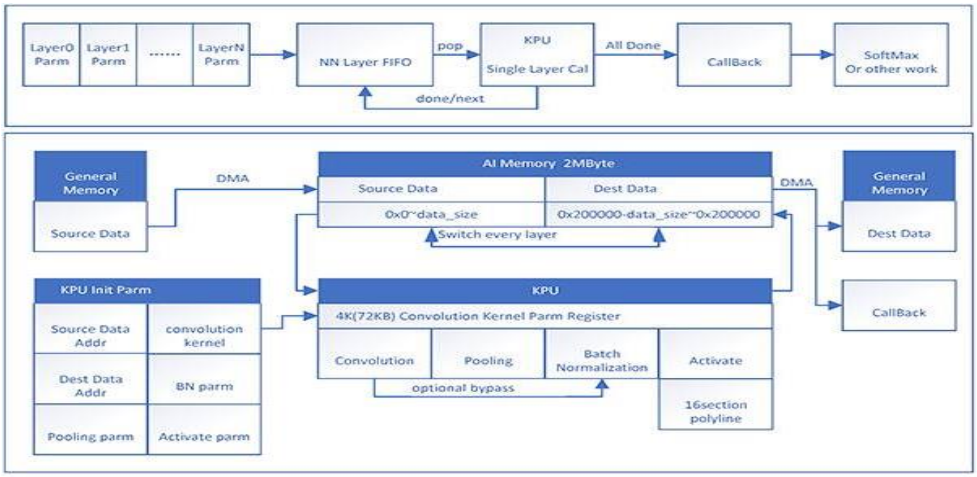
– Chip controlador: K210
– Memoria: 6 MB de memoria de uso general + 2 MB de memoria AI
– Flash: 16 MB
– Soporte MicroPython
– Soporte de depuración: interfaz UART y JTAG
– Acelerómetro digital de 3 ejes: interfaz I2C
– RTC: cristal incorporado de 32.768K conectado con STM32F103
– Interfaz GPIO: todos los GPIO conectados al encabezado 2 * 20P 2.54 mm
– Micrófono MEMS integrado
– Ranura para tarjeta TF a bordo
– Conector de matriz de micrófono
– Puerto de batería de litio
– Puerto USB tipo C
– Chip de codificación de audio + amplificador de audio de doble canal
– Altavoz pequeño 1W
– Botón de marcación de tres vías
– LCD de 2.8 "con pantalla táctil resistiva, resolución 320 * 240

Una de las características más especiales de este prototipo son los aceleradores y la funcionalidad integrada del procesador. Como principal ventaja es la unidad de procesador de audio (APU) admite hasta ocho micrófonos e incluye su propio acelerador FFT de 512 puntos dedicado. Utilizando solo estas capacidades los desarrolladores pueden usar eficientemente conjuntos de micrófonos para implementar

la captación direccional de audio utilizada en interfaces de voz. Para las capacidades de la interfaz de voz, como la activación de frases clave, los desarrolladores pueden usar la salida de audio pre procesada de la APU para controlar el acelerador CNN integrado del procesador. (Evanczuk, 2019).

El método de funcionamiento del microcontrolador y todos su periféricos trabajan capa por capa, la KPU lee los parámetros del modelo y los datos de su SRAM en el chip o memoria SD externa y ejecuta la función del kernel Integrado en cada capa de procesamiento de datos , posee además un mecanismo de devolución de llamada que permite a los desarrolladores ejecutar sus propias interrupciones a medida que el hardware de KPU completa cada secuencia de procesamiento, existe además el procesamiento de aplicaciones de audio al poseer un bus de datos DMA, que esté específicamente designado para realizar interrupciones sin que el microcontrolador realice algún cambio de estado y permita que este siga con los procesos de comunicación y comparación .Finalmente el KPU ejecuta modelos de inferencia utilizando un buffer de entrada que lo usa para procesar secuencialmente cada capa de un modelo de algoritmos. Método que explica (Evanczuk, 2019) en el diagrama de bloques que se visualiza en la figura (2.4).

Figura 2. 4 Método de funcionamiento del MICROCONTROLADOR KPU K210



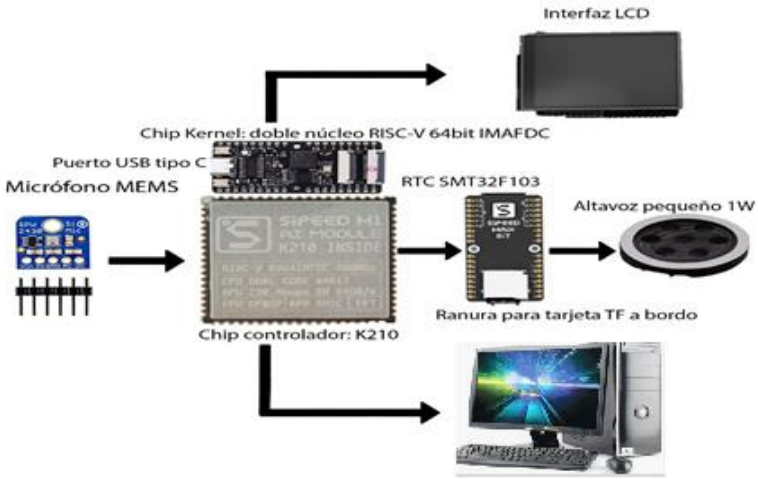
Al realizar la inferencia, la tarea completa de KPU (arriba) comprende múltiples capas, cada una de las cuales involucra la ejecución de las funciones apropiadas del núcleo (abajo). (SIPEED, 2018).

CAPÍTULO 3
DESARROLLO DE PROGRAMACION DEL PROYECTO

3.1 ARQUITECTURA DEL MÉTODO PROPUESTO

Para cumplir a cabalidad con los objetivos citados con anterioridad, se implementará un prototipo electrónico para el reconocimiento de voz, utilizando un microcontrolador K210 como procesador central del circuito electrónico Maix Go. Para sensar los comandos de voz se utilizará un módulo de audio con alta sensibilidad el I2S APU (procesador de audio), el cual se lo conecta a una entrada del prototipo Maix Go. Una vez procesada la señal de audio se emite una respuesta audible a través del módulo reproductor IMAFDC RISC-V al enviarle una señal desde el microcontrolador por comunicación serial. De la misma manera que se envía una señal por el puerto serial al reproductor de audio, se envía una señal a una PC y a la interfaz de LCD que mostrará el nombre del audio que va a reproducir luego de haber reconocido la palabra en el microprocesador. La arquitectura de conexión de los elementos antes mencionados se representa de forma gráfica en la figura (3.1).

Figura 3. 1 Arquitectura de conexión



Esquema de la conexión entre los dispositivos electrónicos utilizados. Fuente: Cusco, J., & León , J. (s.f.).

Para capturar los comandos de voz se utiliza el módulo de Micrófono MEMS (Micro Electrical Mechanical System), integrado al Chip Kernel de doble núcleo RISC-V 64bit IMAFDC de Audio, ya que sus características técnicas explican, que cuenta con un control de ganancia automático posibilitando la adquisición de audio de alta fidelidad en dispositivos electrónicos, portátiles profesionales y de bajo costo. Sin embargo, antes de utilizarlo se realizan varias pruebas con el micrófono KY-038 de Arduino y el modulo MAX 9814, ya que estos módulos de audio son los de más bajo costo que se existen en el mercado. Después de realizar pruebas con ambos se comprueba que el funcionamiento era muy parecido, pero con la diferencia de que el micrófono KY-038 era más sensible al ruido ambiente, lo que no era muy conveniente para el presente proyecto y el Max 9814 es demasiado lento en el tiempo de procesamiento de los comandos de voz, que el prototipo necesita al momento de la evaluación de las tasas de reconocimiento y la generación de las matrices generadas por las características de MEL, el micrófono incorporado al prototipo lo observamos en la figura (3.2.). micrófono

Figura 3. 2 Micrófono MEMS integrado al Chip Kernel.

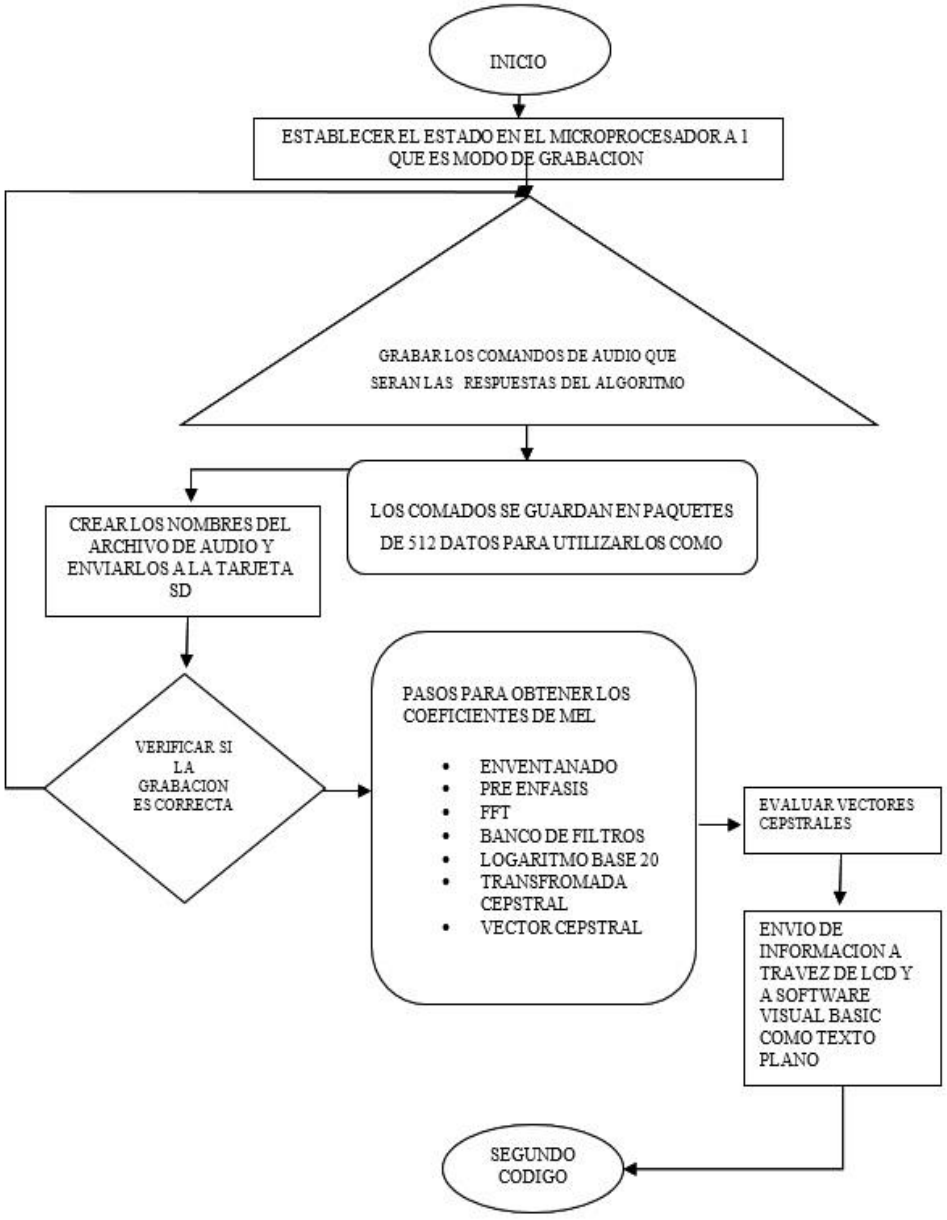
MICROFONO MEMS INTEGRADO AL CHIP KERNEL

Comparación entre los módulos de micrófonos experimentados. (DFROBOT, 2019).

Para realizar el algoritmo de reconocimiento entre comandos de audio y obtener la respuesta audible pregrabada en el prototipo electrónico, se realizan por separado dos códigos de programación en el programa Visual Studio Code, que tiene un entorno de desarrollo integrado llamado PlatformIO IDE, que es el Sistema de compilación multiplataforma usado en Visual Studio, el cual nos permite compilar, grabar y probar con varios tipos de lenguaje de programación como los son C ++, C #, Python, PHP, Go y tiempos de ejecución como NET y Unity. Para poder cumplir con los objetivos

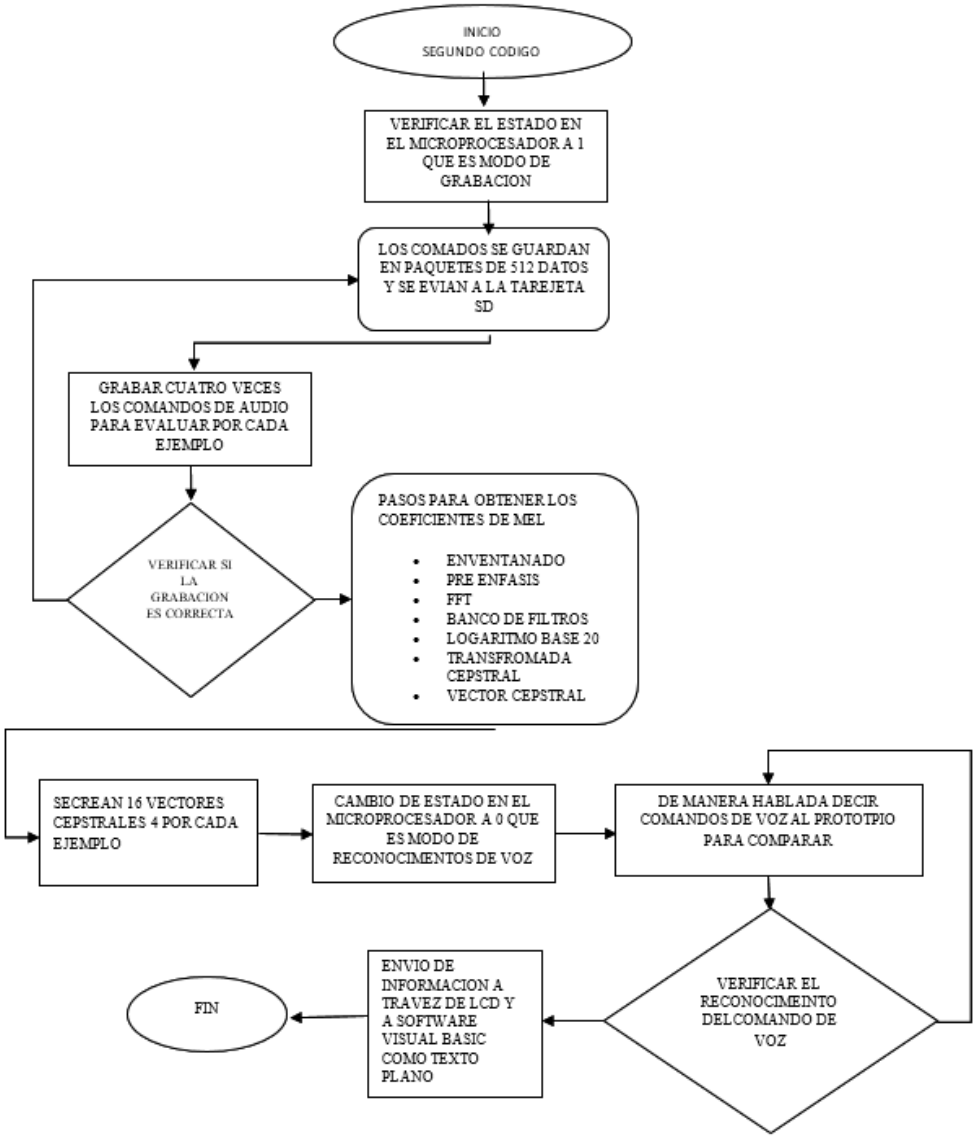
planteados en este prototipo se ha implementado un método con el cual se obtienen resultados satisfactorios cumpliendo un procedimiento que se expone en el siguiente diagrama de flujo representado en la siguiente figura (3.3), dividida en dos partes código de programación 1 y 2.

Figura 3. 3 Representación del diagrama tipo flujo del algoritmo implementado
CODIGO DE PROGRAMACIÓN 1



Representación gráfica mediante diagrama de flujo de los pasos que realiza el primer algoritmo creado.
Elaborado por: Jhon Cusco y John León

Figura 3. 4 Diagrama de flujo del algoritmo implementado CODIGO DE PROGRAMACION 2



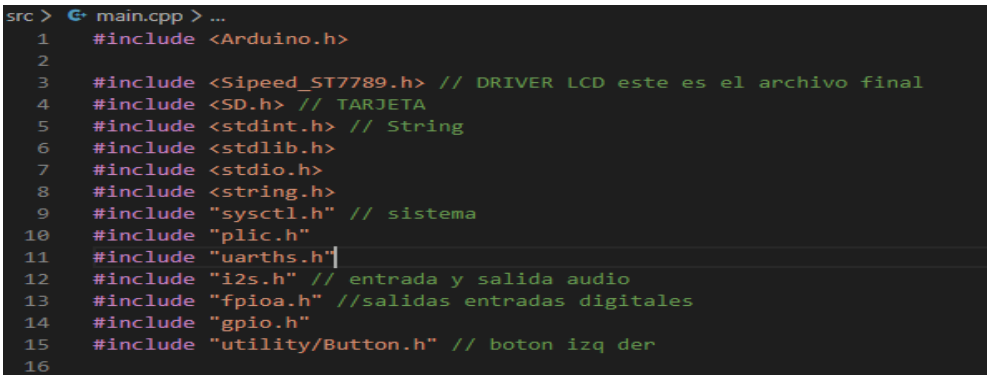
Representación gráfica mediante diagrama de flujo de los pasos que realiza el segundo algorito creado. Elaborado por: Jhon Cusco y John León.

3.2 CÓDIGO DE PROGRAMACIÓN PARA GRABAR COMANDOS DE RESPUESTA

El primer código de programación está estructurado únicamente para almacenar varios comandos de voz, que serán los comandos a reproducir en el altavoz de 1W de amplificación que el prototipo Maix Go tiene ensamblado, este programa activa el micrófono, envía la información escuchada hacia la tarjeta SD y reproduce en altavoz.

Como primer paso para comenzar a grabar en el microcontrolador se debe importar varias librerías como se puede visualizar en la siguiente figura (3.5).

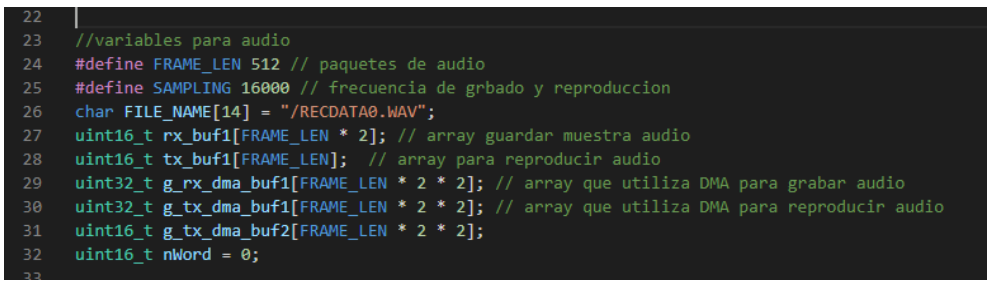
Figura 3. 5 Importación de librerías a Visual Studio



Importación de varias libreras con las que se trabajaran, como las librerías de LCD, tarjeta SD, manejos de String, entrada y salida de audio i2s y la activación de la GPIO a PLATFORMIO. Elaborado por: Jhon Cusco y John León.

Se definen varias variables y tamaños para la visualización que la LCD mostrara, el alto, el ancho y las líneas en donde estarán ubicadas las palabras a mostrar, variables del audio como el FRAME_LEN 512, definido en este valor ya que el de audio se graba por paquetes, en este caso se almacenan 512 datos a una frecuencia de 16 kHz, definida en la variable #define SAMPLING 16000, frecuencia con la que graba y reproduce este prototipo, la cual está establecida en el data sheet de Maix GO y que posee aceleradores que no permitirán que el microprocesador se desborde, porque al muestrear una palabra el Sampling envía 16000 datos que la memoria del microcontrolador no podrá procesar, razón por la cual se graba por paquetes como se aprecia en la figura (3.6).

Figura 3. 6 Variables de audio



Declaración de variables y tamaños de paquetes con los que trabaja el microcontrolador frecuencia de Sampling y tamaños de paquetes. Elaborado por: Jhon Cusco y John León.

Seguido a esto se deberán configuran arreglos en las variables ARRAYS RX y TX, que son los que graban y reproducen, los declaramos para utilizarlos en el bus DMA que permite una trasmisión de datos acelerada, sin que el microcontrolador actúe en el procesamiento, el audio se procesa más rápido en este bus de transmisión, ya que las funciones permiten que se envíe dato a dato el archivo de reproducción hacia la GPIO, para que el microprocesador no realice el proceso de envío como lo hacen los microcontroladores convencionales, programación realizada en la figura (3.7).

Figura 3. 7 Declaración de arreglos para el audio

```
22
23  //variables para audio
24  #define FRAME_LEN 512 // paquetes de audio
25  #define SAMPLING 16000 // frecuencia de grbado y reproduccion
26  char FILE_NAME[14] = "/RECDATA0.WAV";
27  uint16_t rx_buf1[FRAME_LEN * 2]; // array guardar muestra audio
28  uint16_t tx_buf1[FRAME_LEN];  // array para reproducir audio
29  uint32_t g_rx_dma_buf1[FRAME_LEN * 2 * 2]; // array que utiliza DMA para grabar audio
30  uint32_t g_tx_dma_buf1[FRAME_LEN * 2 * 2]; // array que utiliza DMA para reproducir audio
31  uint16_t g_tx_dma_buf2[FRAME_LEN * 2 * 2];
32  uint16_t nWord = 0;
33
34  // banderas de audio, indican procesos de rec y play
35  volatile uint32_t g_index1;
36  volatile uint8_t uart_rec_flag1;
37  volatile uint32_t receive_char1;
38  volatile uint8_t i2s_rec_flag1;
39  volatile uint8_t i2s_start_flag1 = 0;
40
41  //LCD comunicacion SPI
42  SPIClass spi_0(SPI0); // MUST be SPI0 for Maix series on board LCD
43  Sipeed_ST7789 lcd(320, 240, spi_0);
```

Arrays de envío y recepción de datos por el bus DMA que trabaja independientemente del microprocesador para el envío y recepción de audio. Elaborado por: Jhon Cusco y John León.

Para crear la estructura que empiece la grabación de los comandos de voz, se crean las variables del estado en el que se encuentre el audio, ya sea en modo de reproducción, modo de grabación o modo de paro. Además de la creación del nombre con el que el archivo de audio se guardará en la tarjeta SD, Seguidamente se inicializa la tarjeta SD, la LCD, los pines y se deshabilitaran varias funciones y pines del microcontrolador que tiene conectados a la comunicación i2s, por las que se transmiten los datos de la grabación de audio colocándolos en estado bajo o alto, sea cual sea el caso en el que se necesite enviar datos o recibirlos a través de esta comunicación por cada pin declarado, para grabar será el pin I2S0 y el pin I2S2 será para reproducir o transmitir la información, la configuración de los parámetros de grabación se establecen en

varios parámetros como: modo de reproducción, modo de grabado y modo de paro tal como se puede visualizar en las figuras (3.8) la (3.9) y la (3.10).

Figura 3. 8 Estados para la estructura del Audio

```
//definir estructura estado de audio
typedef enum _rec_play_mode
{
    MODE_PLAY = 0,
    MODE_REC = 1,
    MODE_STOP = 2,
    MODE_PLAYING = 3,
    MODE_RECORDING = 4
} rec_play_mode_t;
```

Estados de la estructura en los que se encuentra cada audio grabado ya sea estado de reproducción de audio, grabado de audio y estado de paro. Elaborado por: Jhon Cusco y John León.

Figura 3. 9 Pines establecidos del microcontrolador

```
//declaracion de pines para grabar audio
fpioa_set_function(MIC_DAT3, FUNC_I2S0_IN_D0);
fpioa_set_function(MIC_WS, FUNC_I2S0_WS);
fpioa_set_function(MIC_BCK, FUNC_I2S0_SCLK);

//declaracion de pines para reproducir audio
fpioa_set_function(I2S_DA, FUNC_I2S2_OUT_D1);
fpioa_set_function(I2S_WS, FUNC_I2S2_WS);
fpioa_set_function(I2S_BCK, FUNC_I2S2_SCLK);
```

Declaración de los pines que se van a utilizar en el prototipo para grabar y reproducir los audios IS2 para reproducir y el pin IS0 para grabar. Elaborado por: Jhon Cusco y John León.

Figura 3. 10 Funciones establecidas en la comunicación I2S

```
//i2s init
i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x03);
i2s_init(I2S_DEVICE_2, I2S_TRANSMITTER, 0xC);
```

Las funciones establecidas para la comunicación I2S con el bus de datos de audio DMA al momento de detectar la interrupción guardara la información a transmitir a la tarjeta SD. Elaborado por: Jhon Cusco y John León.

Las variables que se usan para los arreglos de grabación están definidos en DEVICE_0 y para la reproducción en DEVICE_2, para que los buses y los periféricos sepan

cuando actuar según las configuraciones asignadas a las interrupciones, entonces el DMA realizará las interrupciones según la frecuencia ya establecida por el Sampling, estas banderas no interrumpirán ningún proceso en el microcontrolador ya que siempre estarán enviando datos por este bus destinado solo a procesos de audio y realizará las órdenes ya configuradas en cada función. Las funciones que tiene configurado cada interrupción se puede apreciar en la figura (3.11).

Figura 3. 11 Algoritmo para grabación de audio

```
//lo que se hace cuando se interrumpe por DMA / 16000 veces por segundo
//algoritmo de grabado por paquetes
int i2s_dma_irq1(void *ctx) {
    uint32_t i;
    if (i2s_start_flag1) {
        int16_t s_tmp;
        if (g_index1) {
            //recibe datos por DMA y los guarda en los array previamente creados
            i2s_receive_data_dma(I2S_DEVICE_0, &g_rx_dma_buf1[g_index1], FRAME_LEN * 2, DMAC_CHANNEL0);
            g_index1 = 0;
            for (i = 0; i < FRAME_LEN; i++) {
                s_tmp = (int16_t)(g_rx_dma_buf1[2 * i] & 0xffff); //g_rx_dma_buf[2 * i + 1] Low left
                rx_buf1[i] = s_tmp + 32768;
            }
            i2s_rec_flag1 = 1;
        } else {
            i2s_receive_data_dma(I2S_DEVICE_0, &g_rx_dma_buf1[0], FRAME_LEN * 2, DMAC_CHANNEL0);
            g_index1 = FRAME_LEN * 2;
            for (i = FRAME_LEN; i < FRAME_LEN * 2; i++) {
                s_tmp = (int16_t)(g_rx_dma_buf1[2 * i] & 0xffff);//g_rx_dma_buf[2 * i + 1] Low left
                rx_buf1[i] = s_tmp + 32768;
            }
            i2s_rec_flag1 = 2;
        }
    } else {
        i2s_receive_data_dma(I2S_DEVICE_0, &g_rx_dma_buf1[0], FRAME_LEN * 2, DMAC_CHANNEL0);
        g_index1 = FRAME_LEN * 2;
    }
    return 0;
```

Algoritmo de grabación de audio guardando por paquetes en el arreglo DEVICE_0 que es el arreglo de grabado. Elaborado por: Jhon Cusco y John León.

Así como se establecen los parámetros de configuración mediante código, los arreglos, frecuencias, variables y funciones también se establecen las configuraciones de hardware para indicar la acción a realizar manualmente, al dar la orden la persona operaria mediante los botones que tiene incorporado el prototipo se asignaran funciones a realizar para el estado de grabación y reproducción de los comandos de voz que se guardarán en la tarjeta SD, los dos estados del botón serán configurados como pines de control que se puede observar en la figura (3.12).

Figura 3. 12 Distribución de botones del prototipo

```
// disable audio PA
fpioa_set_function(AUDIO_PA_EN_PIN, FUNC_GPIO1);
gpio_set_drive_mode(1, GPIO_DM_OUTPUT);
gpio_set_pin(1, GPIO_PV_LOW);

//declaracion de pines para grabar audio
fpioa_set_function(MIC_DAT3, FUNC_I2S0_IN_D0);
fpioa_set_function(MIC_WS, FUNC_I2S0_WS);
fpioa_set_function(MIC_BCK, FUNC_I2S0_SCLK);

//declaracion de pines para reproducir audio
fpioa_set_function(I2S_DA, FUNC_I2S2_OUT_D1);
fpioa_set_function(I2S_WS, FUNC_I2S2_WS);
fpioa_set_function(I2S_BCK, FUNC_I2S2_SCLK);

//i2s init
i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x03);
i2s_init(I2S_DEVICE_2, I2S_TRANSMITTER, 0xC);

//definir parametros para comunicacion de grabado
i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0,
        RESOLUTION_16_BIT, SCLK_CYCLES_32,
        TRIGGER_LEVEL_4, STANDARD_MODE);

//definir parametros para comunicacion de reproducir
i2s_tx_channel_config(I2S_DEVICE_2, I2S_CHANNEL_1,
        RESOLUTION_16_BIT, SCLK_CYCLES_32,
        TRIGGER_LEVEL_4, RIGHT_JUSTIFYING_MODE);
```

Los botones del prototipo son configurados para que al dar la orden de grabado empiece la interrupción del pin I2S0 y lo guarde en el arreglo DEVICE_0 mediante comunicación I2S. Elaborado por: Jhon Cusco y John León

La configuración de las funciones a realizar para las dos posiciones que tiene el prototipo, serán configuradas para que cuando el selector se mueva a la izquierda se active la interrupción rec_play_mode en el modo de grabación, que es el establecido por la función MODE_RECORDING, este modo permanecerá en esta etapa hasta que el selector se mueva la derecha para activar la función MODE_STOP, función en la cual el archivo de audio se procesa en paquetes de 512 datos por cada comando de voz y se guarda la información en la tarjeta SD para reproducirla en el momento que se detecte el comando hablado y sea reconocido luego de haberlo procesado y comparado con los comandos establecidos, configuraciones que se pueden verificar en la siguiente figura (3.13).

Figura 3. 13 Función rec_play_mode

```
void loop() {
    //leer boto izquierda
    BtnC.read();

    // si rpecionamos boton izq
    if (BtnC.wasPressed()) {
        if (rec_play_mode == MODE_STOP) {
            // stop -> recording
            try {SD.remove(FILE_NAME);}
            catch (char *str) {printf("%s\n", str);}

            file = SD.open(FILE_NAME, FILE_WRITE);
            if (file) {
                drawRecording();
                rec_play_mode = MODE_RECORDING;
            }
            //Detener la grabacion
        } else if (rec_play_mode == MODE_RECORDING) {
            // recording -> stop
            file.close();
            rec_play_mode = MODE_STOP;
            drawStop();

          if (rec_play_mode == MODE_STOP) {
              // stop -> playing
              file = SD.open(FILE_NAME);
              if (file) {
                  rec_play_mode = MODE_PLAYING;
                  drawPlaying();
```

Configuración de las funciones a realizar por movimientos izquierda o derecha del botón selector en el prototipo MAIX GO. Elaborado por: Jhon Cusco y John León

Si bien los parámetros del microcontrolador ya han sido configurados, al igual que todo lo referente al audio, también se deberán configurar los parámetros a visualizar en la pantalla LCD del prototipo, algunas de las configuraciones de inicialización de la LCD ya fueron creadas al principio de la programación al momento de importar las librerías que nos proporciona el software Visual Studio Code, parámetros como de ancho, altura, color , ubicación del cursor, tamaño del texto, color del texto y la línea en donde empezará la escritura de la caratula a mostrar en ambos códigos programados, ya sea en el código compilado de grabación o en el código de reconocimiento de comandos de voz, para este prototipo se utilizará la misma carátula en ambos códigos de programación, cambiando unicamente la linea en donde se mostrará el resultado del comando reconocido y el título de la grabación comparada, configuraciones que se pueden verificar en la captura de la figura (3.14).

Figura 3. 14 Parámetros para la pantalla LCD

```
// Funcion para graficar caratula
void drawMenu() {
    lcd.fillScreen(COLOR_BLACK);
    lcd.fillRect(0, 0, 320, 20, COLOR_CYAN);
    lcd.setCursor(130, 1);
    lcd.setTextSize(2);
    lcd.setTextColor(COLOR_BLACK);
    lcd.println("U.P.S");
    lcd.setCursor(0, 16);
    lcd.println("");
    lcd.setTextColor(COLOR_WHITE);
    for (int i = 0; i < 8; i++) lcd.println("");
    lcd.println("IZQ: REC/PLAY");
    lcd.println("");
    lcd.println("BY:  JHON CUSCO");
    lcd.println("     JOHN LEON");
    lcd.setTextColor(COLOR_GREENYELLOW);
    lcd.print("MAYO 2020");
}
```

Parametros que se pueden configurar en el microcontrolador incorporado en la tarjeta LCD e impresión de la caratula demostrativa del prototipo en funicionamiento. Elaborado por: Jhon Cusco y John León

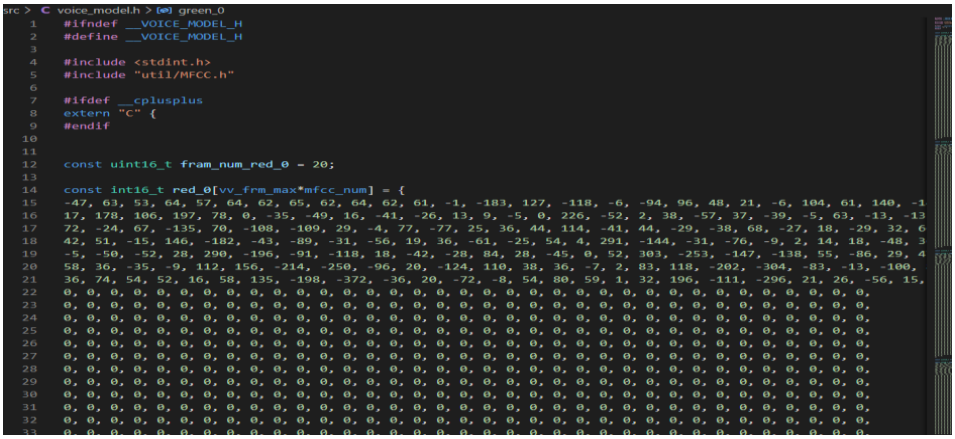
3.3 CÓDIGO DE PROGRAMACIÓN PARA ENTRENAMIENTO DE COMANDOS DE VOZ(ALGORITMO DE RECONOCIMIETO)

Tal como sucede en el oído humano, la forma de identificar audio se lo realiza mediante el dominio de las frecuencias de MEL. Entre las varios métodos de identificación de comandos de voz , la más empleada y eficaz es obtención de coeficientes MFCC de MEL, que es una técnica de adquisición de paquetes de audio, que utilizan los diferentes métodos automáticos para reconocimiento de audio, ya que ofrece una robustez grande al momento de filtrar ruido, característica que hace que este método sea más eficaz a comparación de los otros que tienen problemas significativos al procesar palabras habladas, ya que está basado en un modelo matemático que representa un espectro de potencia de corto plazo en un audio, el cual se basa en una transformación del coseno tipo lineal de un espectro en la potencia de tipo logarítmica, establecida en una escala en la frecuencias de MEL, que se revisó en el apartado 2.3, coeficientes que poseen las características extraídas de los comandos de voz que escucha el micrófono incorporado a la tarjeta GPIO estudiada en el capítulo 2. Para realizar este proceso de captura de datos por coeficientes Cepstrales de MEL se realizaron los siguientes pasos descritos a continuación:

1. Se realiza la transformada de Fourier de un extracto en ventana de una señal.
2. Se realiza el Mapeo de las potencias del espectro obtenido anteriormente en la escala MEL, usando ventanas superpuestas triangulares.
3. Tomando la transformada discreta del coseno de la lista de potencias de registro de MEL, como si fuera una señal.
4. Los MFCC son las amplitudes del espectro resultante.

En la segunda parte de la programación al igual que en la primera se configuran todos los parámetros utilizados para grabar comandos de voz, con la diferencia de que ahora estas grabaciones van a pasar primero por la función con la que obtendremos los coeficientes de MEL, comandos que serán grabados cuatro veces por cada ejemplo, muestra suficiente para realizar la comparación de características obtenidas por cada comando. Para esta etapa de programación que realizara este proceso de adquisición de datos existen dos partes que se adecuarán según el procedimiento que se desea realizar, para comenzar con el entrenamiento de los comandos de voz y sus procesos para adquirir datos con el algoritmo implementado se cambiará el estado de la función RECORD_MODE a 1 estado en el que los comandos de voz se graban y se guardarán los datos obtenidos en otra pestaña del editor llamada Voice Model, en donde estarán todas algunas de las características de los audios obtenidos con el algoritmo, como se representa de forma gráfica en la figura (3.15)

Figura 3. 15 Pestaña Voice Model de Visual Basic



La pestaña Voice Model del software Visual Basic donde están almacenados comando por comando repetido cuatro veces y representado por sus características obtenidas por MEL. Elaborado por: Jhon Cusco y John León

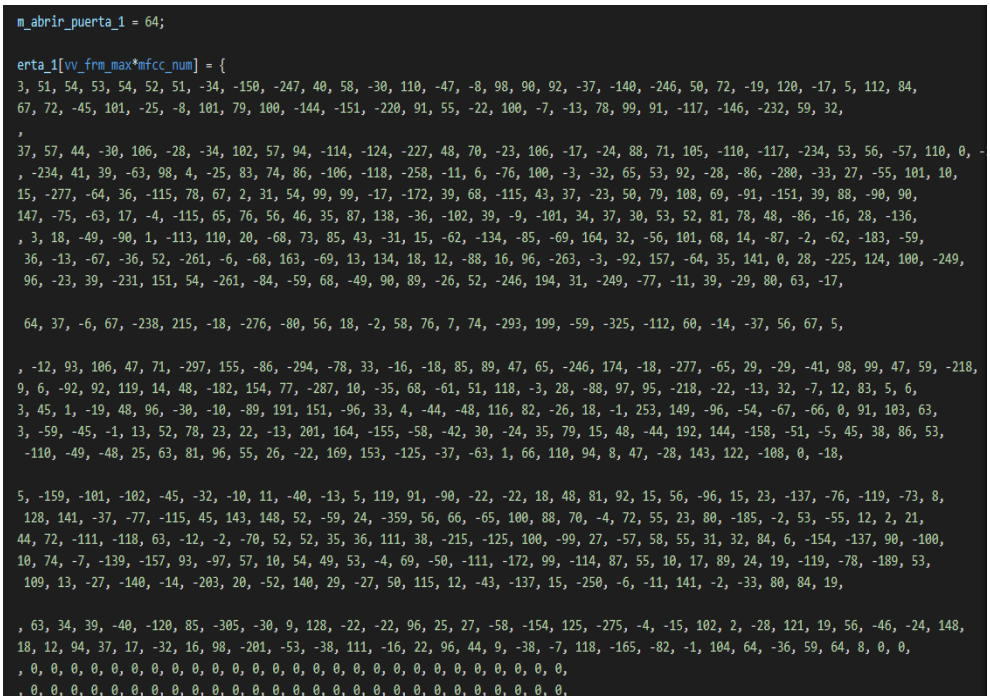
Fueron capturados cuatro ejemplos de comandos de voz, que son: ABRIR PUERTA, BUSCAR MESA, MOVER SILLA Y ENCENDER EQUIPO, comandos de los cuales sus características de audio son obtenidas por los coeficientes de MEL están registrados y guardados en el microcontrolador, comandos dictados verbalmente por una persona de 28 años de edad con tono de voz normal, en un espacio cerrado, controlado del ruido exterior y exceso de eco producido en el ambiente, características ideales para lograr una grabación y un reconocimiento eficaz, características con las que el algoritmo será capaz de evitar confusiones en la adquisición de los coeficientes numéricos, que son extraídos por el algoritmo, evitando adquirir datos de ruido y sonidos producidos por el hablante, que no son necesarios para la comparación de datos de los comandos de voz. Características que se muestran en las siguientes figuras (3.16),(3.17),(3.18).

Figura 3. 16 Comando ABRIR PUERTA GRABACION 1

```
m_abrir_puerta_0 = 67;

erta_0[vv_frm_max*mfcc_num] = {
8, 46, 48, 47, 49, 46, 46, -54, -34, -14, 73, 62, 0, 86, 40, 47, 61, 50, 42, -18, -129, -240, 61, 101, -78, 65, -27, 64, 24, 96,
1, 90, -80, 61, -56, 62, 18, 98, 57, -37, -142, -266, 79, 100, -75, 84, -43, 55, 25, 101, 72, -44, -152, -270, 85, 97, -62,

68, 84, 97, -68, 88, -43, 20, 15, 102, 51, -85, -139, -247, 107, 93, -57, 87, -24, 40, 3, 102, 59, -88, -147, -283, 84, 75,

 -288, 106, 90, -92, 111, -17, 26, 8, 119, 85, -19, -155, -302, 82, 92, -110, 79, -14, 27, 0, 101, 68, -6, -146, -314, 48, 99, -108, 69, 0,
108, -340, 0, 113, -129, 66, 15, 53, -46, 50, 77, 101, -88, -333, -5, 51, -126, 77, 38, 42, -3, 42, 61, 154, -55, -281, 17,

9, -1, -194, 134, 55, -187, 37, 21, 29, 28, 56, 94, 119, -49, -105, 160, 32, -144, 32, 74, 73, -28, 44, 69, 217, -58, -168,

7, 202, -17, -119, 57, 40, -95, 42, 28, 34, -17, 84, 89, 142, 71, -81, 20, 17, -127, 40, 4, 54, -9, 82, 76, 81, 64, -45, 1,

 45, 62, 89, -66, -72, -41, 10, -152, 119, 38, 17, 20, 31, 80, 55, -66, -113, -155, 1, -114, 179, 15, -6, 25, 57, 35, 23, -80, -44, -263,
, 19, 57, 12, 6, -53, 79, -284, 21, -75, 135, 60, 5, 63, 52, 35, -83, 39, 21, -238, 49, -130, 132, -51, 66, 54, 33, 43, -120, 62, 38,
, 64, 99, -6, 47, -184, 106, 3, -244, -60, -97, 32, -59, 108, 73, -23, 21, -178, 119, 5, -260, -47, -35, -5, -36, 86, 70, -17, 70, -193, 144,
, -4, 83, 27, -8, 70, -193, 155, -63, -303, 4, 18, -71, 19, 72, 38, 8, 72, -222, 161, -103, -291, 22, -5, -74, 32, 93, 63, 44, 88, -238,
21, -92, 52, 92, 73, 50, 90, -206, 140, -80, -279, 19, -10, -79, 40, 65, 54, 19, 67, -210, 118, -78, -260, 11, 2, -85, 17, 63, 36, 2, 66,
-6, -5, -46, 21, 108, 43, 22, 62, -194, 85, -50, -269, -39, -43, -26, -25, 128, 57, -13, 79, -82, 116, -27, -266, -29, -85,

, -110, -50, -65, 54, 77, 97, 9, 65, 61, 185, 34, -260, -42, -33, -122, 63, 135, 75, 15, 54, 54, 162, 23, -217, 4, -67, -66, 45, 95, 58,
4, 32, -10, -89, 49, 88, 26, 21, 68, -73, 123, 65, -181, 35, 58, 12, 88, 59, 54, 2, 53, -39, 207, 76, -193, -11, 24, -20, 26, 77,
 135, -148, -34, 7, 10, 10, 80, -10, 10, 50, 11, 198, 109, -96, -65, -4, -40, 46, 85, 30, 19, 41, 3, 165, 63, -89, -24, -44, 13, 101,
12, 69, -63, -7, 17, 6, 47, 68, 53, 52, 62, -42, 42, 29, -14, 17, 2, 103, 70, 69, 65, 30, 49, -6, 77, 43, -28, -49, -44, 66,
4, 120, 48, -6, -21, -17, 40, 35, 65, 57, 55, 33, -336, -68, 14, -84, 39, 31, 12, 22, 103, 3, 117, -10, 37, 38, 37, -61, 55, -13,
, -4, 255, 58, -200, -99, 91, -46, -16, -84, 92, 0, 31, 2, 201, 32, -302, -125, 128, -121, 0, -86, 117, 19, 16, 23, 180, 27, -282, -129, 142,
, 8, 27, 148, -13, -282, -140, 174, -172, 32, -6, 109, -1, -3, 25, 83, -74, -207, -144, 130, -162, 104, 5, 37, -33, 61, 41,
```

Características extraídas por algoritmo de reconocimiento de voz se detectan 1948 datos. Elaborado por: Jhon Cusco y John León.

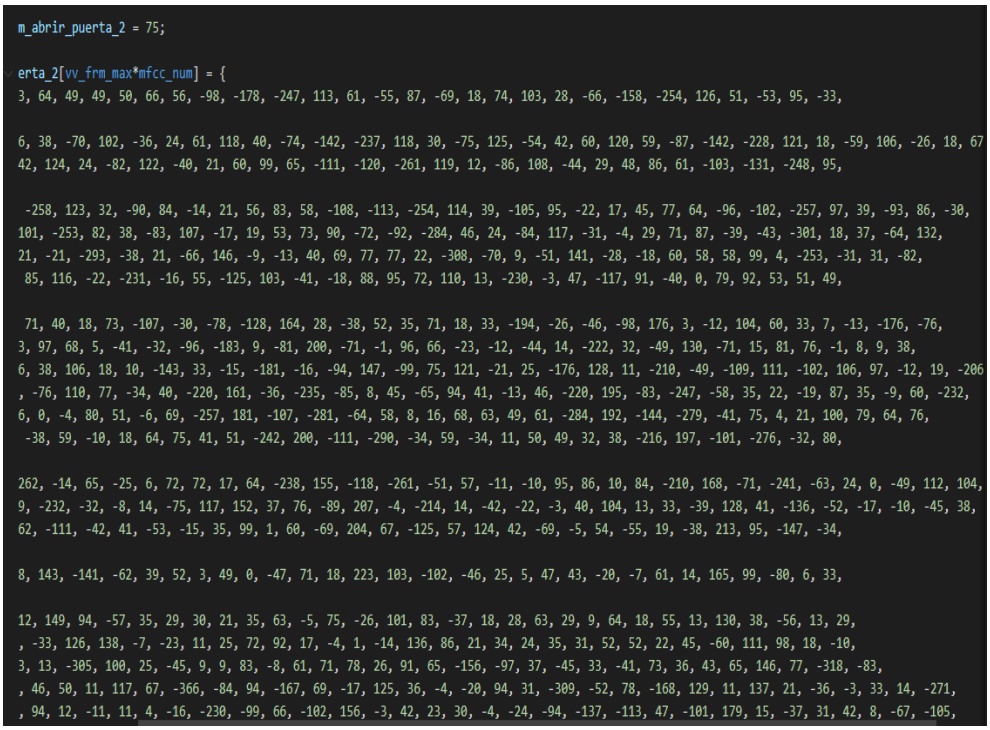
Figura 3. 17 Comando ABRIR PUERTA GRABACION 2



```
m_abrir_puerta_1 = 64;

erta_1[vv_frm_max*mfcc_num] = {
3, 51, 54, 53, 54, 52, 51, -34, -150, -247, 40, 58, -30, 110, -47, -8, 98, 90, 92, -37, -140, -246, 50, 72, -19, 120, -17, 5, 112, 84,
67, 72, -45, 101, -25, -8, 101, 79, 100, -144, -151, -220, 91, 55, -22, 100, -7, -13, 78, 99, 91, -117, -146, -232, 59, 32,
,
37, 57, 44, -30, 106, -28, -34, 102, 57, 94, -114, -124, -227, 48, 70, -23, 106, -17, -24, 88, 71, 105, -110, -117, -234, 53, 56, -57, 110, 0, -
, -234, 41, 39, -63, 98, 4, -25, 83, 74, 86, -106, -118, -258, -11, 6, -76, 100, -3, -32, 65, 53, 92, -28, -86, -280, -33, 27, -55, 101, 10,
15, -277, -64, 36, -115, 78, 67, 2, 31, 54, 99, 99, -17, -172, 39, 68, -115, 43, 37, -23, 50, 79, 108, 69, -91, -151, 39, 88, -90, 90,
147, -75, -63, 17, -4, -115, 65, 76, 56, 46, 35, 87, 138, -36, -102, 39, -9, -101, 34, 37, 30, 53, 52, 81, 78, 48, -86, -16, 28, -136,
, 3, 18, -49, -90, 1, -113, 110, 20, -68, 73, 85, 43, -31, 15, -62, -134, -85, -69, 164, 32, -56, 101, 68, 14, -87, -2, -62, -183, -59,
 36, -13, -67, -36, 52, -261, -6, -68, 163, -69, 13, 134, 18, 12, -88, 16, 96, -263, -3, -92, 157, -64, 35, 141, 0, 28, -225, 124, 100, -249,
 96, -23, 39, -231, 151, 54, -261, -84, -59, 68, -49, 90, 89, -26, 52, -246, 194, 31, -249, -77, -11, 39, -29, 80, 63, -17,

 64, 37, -6, 67, -238, 215, -18, -276, -80, 56, 18, -2, 58, 76, 7, 74, -293, 199, -59, -325, -112, 60, -14, -37, 56, 67, 5,

, -12, 93, 106, 47, 71, -297, 155, -86, -294, -78, 33, -16, -18, 85, 89, 47, 65, -246, 174, -18, -277, -65, 29, -29, -41, 98, 99, 47, 59, -218,
9, 6, -92, 92, 119, 14, 48, -182, 154, 77, -287, 10, -35, 68, -61, 51, 118, -3, 28, -88, 97, 95, -218, -22, -13, 32, -7, 12, 83, 5, 6,
3, 45, 1, -19, 48, 96, -30, -10, -89, 191, 151, -96, 33, 4, -44, -48, 116, 82, -26, 18, -1, 253, 149, -96, -54, -67, -66, 0, 91, 103, 63,
3, -59, -45, -1, 13, 52, 78, 23, 22, -13, 201, 164, -155, -58, -42, 30, -24, 35, 79, 15, 48, -44, 192, 144, -158, -51, -5, 45, 38, 86, 53,
 -110, -49, -48, 25, 63, 81, 96, 55, 26, -22, 169, 153, -125, -37, -63, 1, 66, 110, 94, 8, 47, -28, 143, 122, -108, 0, -18,

5, -159, -101, -102, -45, -32, -10, 11, -40, -13, 5, 119, 91, -90, -22, -22, 18, 48, 81, 92, 15, 56, -96, 15, 23, -137, -76, -119, -73, 8,
 128, 141, -37, -77, -115, 45, 143, 148, 52, -59, 24, -359, 56, 66, -65, 100, 88, 70, -4, 72, 55, 23, 80, -185, -2, 53, -55, 12, 2, 21,
44, 72, -111, -118, 63, -12, -2, -70, 52, 52, 35, 36, 111, 38, -215, -125, 100, -99, 27, -57, 58, 55, 31, 32, 84, 6, -154, -137, 90, -100,
10, 74, -7, -139, -157, 93, -97, 57, 10, 54, 49, 53, -4, 69, -50, -111, -172, 99, -114, 87, 55, 10, 17, 89, 24, 19, -119, -78, -189, 53,
 109, 13, -27, -140, -14, -203, 20, -52, 140, 29, -27, 50, 115, 12, -43, -137, 15, -250, -6, -11, 141, -2, -33, 80, 84, 19,

, 63, 34, 39, -40, -120, 85, -305, -30, 9, 128, -22, -22, 96, 25, 27, -58, -154, 125, -275, -4, -15, 102, 2, -28, 121, 19, 56, -46, -24, 148,
18, 12, 94, 37, 17, -32, 16, 98, -201, -53, -38, 111, -16, 22, 96, 44, 9, -38, -7, 118, -165, -82, -1, 104, 64, -36, 59, 64, 8, 0, 0,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Características extraídas por algoritmo de reconocimiento de voz se detectan 1964 datos. Elaborado por: Jhon Cusco y John León.

Figura 3. 18 Comando ABRIR PUERTA GRABACION 3



```
m_abrir_puerta_2 = 75;

erta_2[vv_frm_max*mfcc_num] = {
3, 64, 49, 49, 50, 66, 56, -98, -178, -247, 113, 61, -55, 87, -69, 18, 74, 103, 28, -66, -158, -254, 126, 51, -53, 95, -33,

6, 38, -70, 102, -36, 24, 61, 118, 40, -74, -142, -237, 118, 30, -75, 125, -54, 42, 60, 120, 59, -87, -142, -228, 121, 18, -59, 106, -26, 18, 67
42, 124, 24, -82, 122, -40, 21, 60, 99, 65, -111, -120, -261, 119, 12, -86, 108, -44, 29, 48, 86, 61, -103, -131, -248, 95,

 -258, 123, 32, -90, 84, -14, 21, 56, 83, 58, -108, -113, -254, 114, 39, -105, 95, -22, 17, 45, 77, 64, -96, -102, -257, 97, 39, -93, 86, -30,
101, -253, 82, 38, -83, 107, -17, 19, 53, 73, 90, -72, -92, -284, 46, 24, -84, 117, -31, -4, 29, 71, 87, -39, -43, -301, 18, 37, -64, 132,
21, -21, -293, -38, 21, -66, 146, -9, -13, 40, 69, 77, 77, 22, -308, -70, 9, -51, 141, -28, -18, 60, 58, 58, 99, 4, -253, -31, 31, -82,
 85, 116, -22, -231, -16, 55, -125, 103, -41, -18, 88, 95, 72, 110, 13, -230, -3, 47, -117, 91, -40, 0, 79, 92, 53, 51, 49,

 71, 40, 18, 73, -107, -30, -78, -128, 164, 28, -38, 52, 35, 71, 18, 33, -194, -26, -46, -98, 176, 3, -12, 104, 60, 33, 7, -13, -176, -76,
3, 97, 68, 5, -41, -32, -96, -183, 9, -81, 200, -71, -1, 96, 66, -23, -12, -44, 14, -222, 32, -49, 130, -71, 15, 81, 76, -1, 8, 9, 38,
6, 38, 106, 18, 10, -143, 33, -15, -181, -16, -94, 147, -99, 75, 121, -21, 25, -176, 128, 11, -210, -49, -109, 111, -102, 106, 97, -12, 19, -206
, -76, 110, 77, -34, 40, -220, 161, -36, -235, -85, 8, 45, -65, 94, 41, -13, 46, -220, 195, -83, -247, -58, 35, 22, -19, 87, 35, -9, 60, -232,
6, 0, -4, 80, 51, -6, 69, -257, 181, -107, -281, -64, 58, 8, 16, 68, 63, 49, 61, -284, 192, -144, -279, -41, 75, 4, 21, 100, 79, 64, 76,
 -38, 59, -10, 18, 64, 75, 41, 51, -242, 200, -111, -290, -34, 59, -34, 11, 50, 49, 32, 38, -216, 197, -101, -276, -32, 80,

262, -14, 65, -25, 6, 72, 72, 17, 64, -238, 155, -118, -261, -51, 57, -11, -10, 95, 86, 10, 84, -210, 168, -71, -241, -63, 24, 0, -49, 112, 104,
9, -232, -32, -8, 14, -75, 117, 152, 37, 76, -89, 207, -4, -214, 14, -42, -22, -3, 40, 104, 13, 33, -39, 128, 41, -136, -52, -17, -10, -45, 38,
62, -111, -42, 41, -53, -15, 35, 99, 1, 60, -69, 204, 67, -125, 57, 124, 42, -69, -5, 54, -55, 19, -38, 213, 95, -147, -34,

8, 143, -141, -62, 39, 52, 3, 49, 0, -47, 71, 18, 223, 103, -102, -46, 25, 5, 47, 43, -20, -7, 61, 14, 165, 99, -80, 6, 33,

12, 149, 94, -57, 35, 29, 30, 21, 35, 63, -5, 75, -26, 101, 83, -37, 18, 28, 63, 29, 9, 64, 18, 55, 13, 130, 38, -56, 13, 29,
, -33, 126, 138, -7, -23, 11, 25, 72, 92, 17, -4, 1, -14, 136, 86, 21, 34, 24, 35, 31, 52, 52, 22, 45, -60, 111, 98, 18, -10,
3, 13, -305, 100, 25, -45, 9, 9, 83, -8, 61, 71, 78, 26, 91, 65, -156, -97, 37, -45, 33, -41, 73, 36, 43, 65, 146, 77, -318, -83,
, 46, 50, 11, 117, 67, -366, -84, 94, -167, 69, -17, 125, 36, -4, -20, 94, 31, -309, -52, 78, -168, 129, 11, 137, 21, -36, -3, 33, 14, -271,
, 94, 12, -11, 11, 4, -16, -230, -99, 66, -102, 156, -3, 42, 23, 30, -4, -24, -94, -137, -113, 47, -101, 179, 15, -37, 31, 42, 8, -67, -105,
```

Características extraídas por algoritmo de reconocimiento de voz se detectan 1937 datos. Elaborado por: Jhon Cusco y John León.

3.4 PROCESAR LOS DATOS DE AUDIO

Teniendo ya grabados los comandos de voz que se necesitan reconocer, empezará ya el desarrollo del algoritmo. Al estudiar coeficientes de MEL en el capítulo dos, se explicaron todos los parámetros que realiza este modelo matemático para determinar un valor de retorno y un vector, algoritmo que extrae las características del comando de voz emitido por una persona, que está dentro de una escala de tipo lineal entre los 1000 Hz y en una escala logarítmica por encima de los 1000 Hz, que se aplican para acentuar la forma en la que se adquieren datos en este modelo , dentro de esta escala ya sea comparada en forma lineal o logarítmica. Para extraer características de una palabra se realizan varios procesos que permitirán adquirir datos con la ayuda de dos librerías que son las más importantes en la fase que extraerá varias características de comandos de audio, las dos librerías fueron importadas de la plataforma de programación PLATFORMIO, la librería número uno es la que permite realizar el reconocimiento de los comandos de voz Maix_Speech_Recognition; esta librería es la que permite comparar los datos que fueron adquiridos con la primera programación estudiada en el apartado 3.2, en donde se guardan los comandos de voz que servirán de respuesta y las características extraídas por el algoritmo.

La librería trabaja de igual manera con las mismas funciones y parámetros de inicialización con los que se activa el micrófono y adquiere paquetes de 512 datos, trabaja de igual manera con la función creada device_0 y la device_2 que se revisó en la programación de la figura 3.9 y 3.10 del apartado 3.2 , cambia de estado el micrófono ya sea a modo de grabación o a modo de stop, tomando en cuenta aquí las escalas en las que se descarta si es ruido o son comandos de voz los cuales se deben procesar y adquirir los datos, a continuación se muestra parte del código implementado en el algoritmo de la figura (3.19).

Figura 3. 19 Código de la librería Maix_Speech_Recognition

```
#incluye "Maix_Speech_Recognition.h"
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#incluye "sysctl.h"
#incluye "plic.h"
#incluye "uarths.h"
#incluye "util/g_def.h"
#incluye "i2s.h"
#incluye "fpioa.h"

#incluye "util/VAD.h"
#incluye "util/MFCC.h"
#incluye "util/DTW.h"
#incluye "util/flash.h"
#incluye "util/ADC.h"

#define USART1_printf Serial.printf

uint16_t VcBuf[atap_len];
atap_tag atap_arg;
valid_tag valid_voice[max_vc_con];
v_ftr_tag ftr;
v_ftr_tag ftr_temp;
v_ftr_tag ftr_mdl_temp[10];
v_ftr_tag *pftr_mdl_temp[10];

#define save_ok      0
#define VAD_fail     1
#define MFCC_fail    2
#define Flash_fail   3
```

```
# define  Flash_fail  3

# define  FFT_N  512

uint16_t rx_buf [FRAME_LEN];
uint32_t g_rx_dma_buf [FRAME_LEN * 2 ];
uint64_t fft_out_data [FFT_N / 2 ];

volátil  uint32_t g_index;
volátil  uint8_t uart_rec_flag;
volátil  uint32_t recibir_char ;
volátil  uint8_t i2s_rec_flag;
volátil  uint8_t i2s_start_flag = 0 ;

int  i2s_dma_irq ( void * ctx)
{
    uint32_t i;
    if (i2s_start_flag)
    {
        int16_t s_tmp;
        si (g_index)
        {
            i2s_receive_data_dma (I2S_DEVICE_0, & g_rx_dma_buf [g_index], frame_mov * 2 , DMAC_CHANNEL3);
            g_index = 0 ;
            para (i = 0 ; i <frame_mov; i ++)
            {
                s_tmp = ( int16_t ) (g_rx_dma_buf [ 2 * i] & 0xffff ); // g_rx_dma_buf [2 * i + 1] Baja izquierda
                rx_buf [i] = s_tmp + 32768 ;
            }
            i2s_rec_flag = 1 ;
```

Código de programación de la librería Maix_Speech_Recognition encargada de diferenciar si las lecturas que se están realizando en prototipo son audio o ruido y nos entrega datos guardaos en la variable MFCC.C. (MAIXDUINO, 2019)

En esta parte de la librería se verifica el cambio de estado del modo de grabación en el bus de datos DMA, ya que los arreglos de grabación fueron configurados como DEVICE_0, para adquisición de datos y para la reproducción en DEVICE_2, para que los buses y los periféricos sepan cuando actuar según las configuraciones asignadas a las interrupciones, entonces el DMA realizará las interrupciones según la frecuencia ya establecida por el Sampling, estas banderas no interrumpirán ningún proceso en el microcontrolador ya que siempre estarán enviando datos por este bus destinado solo a procesos audio.

La siguiente parte del código de la librería Maix_Speech_Recognition.C, está ya orientada a determinar si es ruido o es una palabra determinada por varias características propias, ya estudiadas en el apartado 2.2 y el 2.1, donde se establecieron las características del audio, parámetros, niveles segmentos de voz y audio válidos para diferenciar de comandos de voz y ruido, que es lo que se muestra en parte de este código de programación en la figura (3.20).

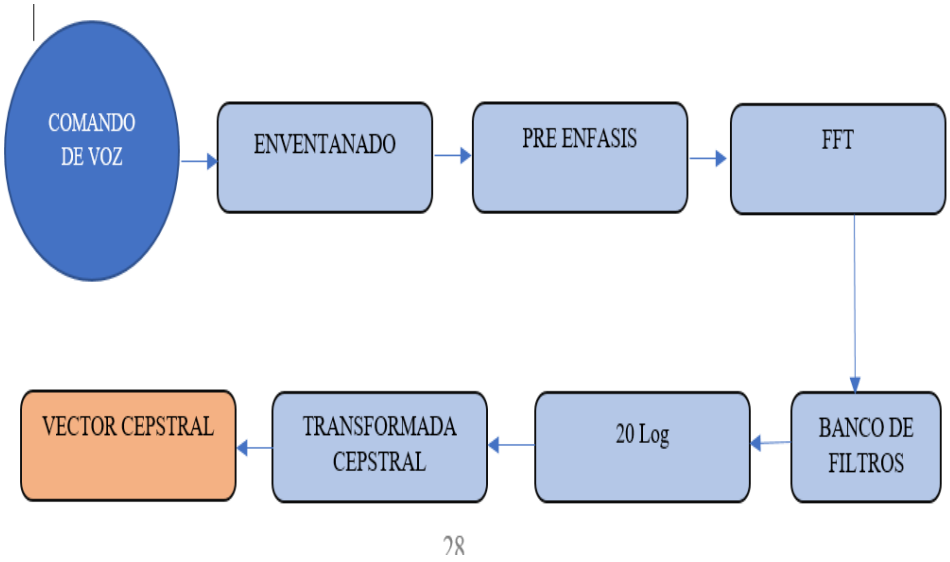
Figura 3. 20 Código de la librería Maix_Speech_Recognition

```
184 uint8_t  SpeechRecognizer :: save_mdl ( uint16_t * v_dat, uint32_t addr)
185 {
186     ui6 i, num;
187     ui6 frame_index;
188 get_noise1:
189     frame_index = 0 ;
190     num = atap_len / frame_mov;
191     // espera a que termine
192     mientras que ( 1 ) {
193         while (i2s_rec_flag == 0 )
194             continuar ;
195         if (i2s_rec_flag == 1 ) {
196             para (i = 0 ; i <frame_mov; i ++)
197                 v_dat [frame_mov * frame_index + i] = rx_buf [i];
198         } más {
199             para (i = 0 ; i <frame_mov; i ++)
200                 v_dat [frame_mov * frame_index + i] = rx_buf [i + frame_mo\
201         }
202         i2s_rec_flag = 0 ;
203         frame_index ++;
204         if (frame_index> = num)
205             romper ;
206     }
207 //   para (i = 0; i <atap_len; i ++)
208 //       Serial.printf ("ruido:% d \ n", v_dat [i]);
209     noise_atap (v_dat, atap_len, y atap_arg);
210     if (atap_arg. s_thl > 10000 ) {
211         De serie. printf ( " obtener ruido de nuevo ... \ n " );
212         ir a get_noise1;
213     }
214     De serie. printf ( " hablando ... \ n " );
215 // espera a que termine
            continuar ;
        if (i2s_rec_flag == 1 ) {
            para (i = 0 ; i <frame_mov; i ++) {
                v_dat [i] = v_dat [i + frame_mov];
                v_dat [i + frame_mov] = rx_buf [i];
            }
        } más {
            para (i = 0 ; i <frame_mov; i ++) {
                v_dat [i] = v_dat [i + frame_mov];
                v_dat [i + frame_mov] = rx_buf [i + frame_mov
            }
        }
        i2s_rec_flag = 0 ;
        if ( VAD2 (v_dat, valid_voice, & atap_arg) == 1 )
            romper ;
        if (recibir_char == ' s ' )
            return MFCC_fail;
    }
//   if (valid_voice [0] .end == ((void *) 0)) {
//       Serial.printf ("VAD_fail \ n");
//       devuelve VAD_fail;
//   }

    get_mfcc (& (valid_voice [ 0 ]), & ftr, & atap_arg);
    if (ftr. frm_num == 0 ) {
        // Serial.printf ("MFCC_fail \ n");
        return MFCC_fail;
```

Código de programación de la librería Maix_Speech_Recognition encargada de diferenciar si las lecturas que se están realizando en prototipo son audio o ruido y nos entrega datos guardaos en la variable MFCC:C. (MAIXDUINO, 2019)

La respuesta que envía este código de programación se guardará en la variable MFCC que serán los datos que se utilizarán en la segunda librería importada y propia del prototipo de MAIX GO, que fue importada con el nombre de MFCC.C, esta librería es la que permitirá extraer todos los datos y convertirlos en vector cepstral el cual extraerá valores de las características propias de cada comando de voz, que se comparan uno a uno, esta librería tiene varios pasos y la secuencia a seguir que fue establecida en la figura 3.3 del apartado 3.1 de manera general, se representa de forma gráfica en la figura el diagrama de bloques del funcionamiento que tiene la librería MFCC.C y se describen sus características de manera gráfica en el diagrama representado por bloques en la figura (3.21)

Figura 3. 21 Diagrama de bloques de librería MFCC



Pasos que sigue uno a uno explicado en diagrama de bloques de la librería de MFCC. Elaborado por: Jhon Cusco y John León

Se analizarán cada una las fases que contiene la imagen 3.21 de manera teórica, basada sobre el software, y se determinarán las características de los valores que se han asignado a las medidas obtenidas en las pruebas para obtener los coeficientes de MEL Cepstrales.

Enventanado es el proceso en el cual los comandos de voz pasan por una fase aleatoria y de manera no estacionaria seria el proceso más complicado al momento de

analizarlos, este proceso es posible solucionarlo trabajando en cortos tiempos de ms, obteniendo así una señal casi-estacionaria, también un análisis que permite obtener segmentos y tramas de las señales con varios ms que se denomina análisis de tipo localizado. El proceso que genera la trama y el segmento consecutivo de una señal se lo conoce como enventanado. Normalmente se trabaja con ventanas llamadas Hamming, que poseen un tamaño de 20 ms. Que son necesarios para mantener continua la información de las señales, comúnmente el enventanado se lo realiza con los bloques de muestras que están solapadas entre ellas, de esta manera no se perderá información en las transiciones entre las ventanas. Normalmente el solapamiento lo realiza desplazándose entre ventanas con un tiempo de 10 ms, logrando obtener coeficientes de MEL a partir de 10 ms, que se representa de forma gráfica en la en la figura (3.22)

Figura 3. 22 Enventanado de coeficientes de MEL

```
    //SACA EL MFCC
    get_mfcc(&(valid_voice[0]), &ftr, &atap_arg);
    if (ftr.frm_num == 0) {
        *mtch_dis = dis_err;
        Serial.printf("MFCC fail ");
        return 0;
    }
//  for (i = 0; i < ftr.frm_num * mfcc_num; i++) {
//      if (i % 12 == 0)
//          Serial.printf("\n");
//      Serial.printf("%d ", ftr.mfcc_dat[i]);
//  }
//  ftr.word_num = valid_voice[0].word_num;
    Serial.printf("mfcc ok\n");
    i = 0;
    min_comm = 0;
    min_dis = dis_max;
    cycle0 = read_csr(mcycle);
    for (ftr_addr = ftr_start_addr; ftr_addr < ftr_end_addr; ftr_addr += size_per_ftr)  {
    //  ftr_mdl=(v_ftr_tag*)ftr_addr;
        ftr_mdl = (v_ftr_tag *)(&ftr_save[ftr_addr / size_per_ftr]);
        cur_dis = ((ftr_mdl->save_sign) == save_mask) ? dtw(ftr_mdl, &ftr) : dis_err;
        if ((ftr_mdl->save_sign) == save_mask) {
            Serial.printf("no. %d, frm_num = %d, save_mask=%d", i + 1, ftr_mdl->frm_num, ftr_mdl->save_sign);
            Serial.printf("cur_dis=%d\n", cur_dis);
        }
```

Código de programación para el cálculo de MFCC y obtención de coeficientes. Elaborado por: Jhon Cusco y John León

Para la fase de Pre-énfasis las señales de la voz se parametrizan y atraviesan un filtro. La cualidad de este proceso es sumar a la atenuación de 20 dB que se producen en esta fase fisiológica al momento de la producción de los comandos de voz, filtro obtenido

mediante código de programación que se representa de forma gráfica en la figura (3.23).

Figura 3. 23 Filtro de Pre-énfasis

```
    //Pre-énfasis
//  printf("vc_dat[%d]=%d ",i,((s32)(*(vc_dat+i))-mid));
    temp = ((s32)(*(vc_dat+i))-mid) - hp_ratio(((s32)(*(vc_dat+i-1))-mid));
//  printf("vc_hp[%d]=%d ",i,temp);
    //Agregue la ventana Hamming y amplíe 10 veces
    vc_temp[i] = (s16)(temp*hamm[i]/(hamm_top/10));
//  printf("vc_hm[%d]=%d\r\n",i,vc_temp[i]);
}

frq_spct = mfcc_fft(vc_temp, FRAME_LEN);

for (i = 0; i < frq_max; i++) {
    //printf("frq_spct[%d]=%d ",i,frq_spct[i]);
    frq_spct[i] *= frq_spct[i];//Espectro de energía
    //printf("E_spct[%d]=%d\r\n",i,frq_spct[i]);
}
```

Código para el cálculo del espectro de energía guardado en la variable temp. Elaborado por: Jhon Cusco y John León

La señal pre enfatizada se obtendrá mediante siguiente filtro representado en la ecuación (3.1).

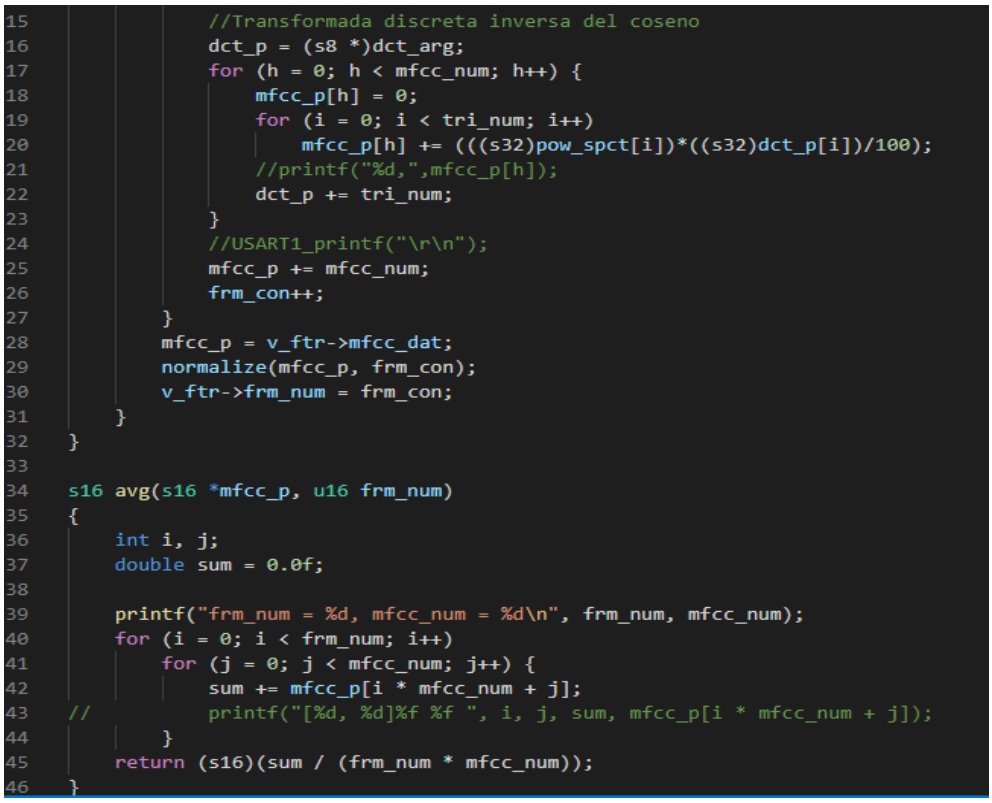
$$y[n] = x[n] - ax[n-1] \qquad \textit{Ec. (3.1)}$$

FFT. Luego de haber realizados estos dos procesos, se procede a calcular la transformada rápida de fourier con tamaño N del enventanado usando la siguiente ecuación (3.2).

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk}, 0 \leq k \leq N \qquad \textit{Ec. (3.2)}$$

Luego de realizar esta operación la fase es descartada, trabajando con la parte envolvente de la señal de audio.$|X[k]|$, operación realizada mediante código de programación observada en la figura (3.24).

Figura 3. 24 Programación para la transformada inversa del coseno



```
            //Transformada discreta inversa del coseno
            dct_p = (s8 *)dct_arg;
            for (h = 0; h < mfcc_num; h++) {
                mfcc_p[h] = 0;
                for (i = 0; i < tri_num; i++)
                    mfcc_p[h] += (((s32)pow_spct[i])*((s32)dct_p[i])/100);
                //printf("%d,",mfcc_p[h]);
                dct_p += tri_num;
            }
            //USART1_printf("\r\n");
            mfcc_p += mfcc_num;
            frm_con++;
        }
        mfcc_p = v_ftr->mfcc_dat;
        normalize(mfcc_p, frm_con);
        v_ftr->frm_num = frm_con;
    }
}

s16 avg(s16 *mfcc_p, u16 frm_num)
{
    int i, j;
    double sum = 0.0f;

    printf("frm_num = %d, mfcc_num = %d\n", frm_num, mfcc_num);
    for (i = 0; i < frm_num; i++)
        for (j = 0; j < mfcc_num; j++) {
            sum += mfcc_p[i * mfcc_num + j];
//          printf("[%d, %d]%f %f ", i, j, sum, mfcc_p[i * mfcc_num + j]);
        }
    return (s16)(sum / (frm_num * mfcc_num));
}
```

Aplicación de la transformada inversa del coseno a través de código de programación. Elaborado por: Jhon Cusco y John León.

Banco de filtros. En este punto del algoritmo la señal esta multiplicada por los filtros triangulares que estarán separados por la escala de frecuencia de MEL. Como ejemplo de esto se muestra la Figura 3.4 del apartado 3.2 en donde está definido cada filtro por su frecuencia central de manera adyacentes. Para esto el diagrama de filtros está conformado por 20 ejemplo entre los cuales, 10 son linealmente separados de100 a 1000 Hz, cinco están logarítmicamente separados de 1 a 2 kHz y los otros cinco logarítmicamente separados de 2 a 4 kHz. Además, el ancho de banda para estos filtros triangulares se determina por la clasificación de las frecuencias centrales de cada uno de los filtros, ahora serán las funciones de las frecuencias de muestreos y de la cantidad de filtros. A medida que la cantidad de filtros sube, se disminuye proporcionalmente el ancho de banda en cada uno de los filtros en la escala de frecuencias de MEL haciendo uso de la ecuación (3.3).

$$mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \qquad \textit{Ec. (3.3)}$$

Donde f representa a las frecuencias en la escala lineal, este proceso se lo realiza en código de programación tal y como se representa gráficamente en la figura (3.25)

Figura 3. 25 Filtros triangulares establecidos a través de código de programación.

```
//Filtro triangular
pow_spct[0] = 0;
for (i = 0; i < tri_cen[1]; i++)
    pow_spct[0] += (frq_spct[i]*tri_even[i]/(tri_top/10));
for (h = 2; h < tri_num; h += 2) {
    pow_spct[h] = 0;
    for (i = tri_cen[h-1]; i < tri_cen[h+1]; i++)
        pow_spct[h] += (frq_spct[i]*tri_even[i]/(tri_top/10));
}

for (h = 1; h < (tri_num-2); h += 2) {
    pow_spct[h] = 0;
    for (i = tri_cen[h-1]; i < tri_cen[h+1]; i++)
        pow_spct[h] += (frq_spct[i]*tri_odd[i]/(tri_top/10));
}
pow_spct[tri_num-1] = 0;
for (i = tri_cen[tri_num-2]; i < (mfcc_fft_point/2); i++)
    pow_spct[tri_num-1] += (frq_spct[i]*tri_odd[i]/(tri_top/10));

//La salida del filtro triangular es logarítmica
for (h = 0; h < tri_num; h++) {
    //USART1_printf("pow_spct[%d]=%d ",h,pow_spct[h]);
    pow_spct[h] = (u32)(log(pow_spct[h])*100);//Después de tomar el logaritmo, multiplique por 100 par
    //USART1_printf("%d\r\n",pow_spct[h]);
}
```

Caculo de filtro triangular con respuesta logarítmica .. Elaborado por: Jhon Cusco y John León.

El modelo matemático para calcular el valor de cada uno de estos filtros está determinado por la siguiente ecuación (3.4)

$$H_{m[k]} = \frac{\frac{0}{\frac{2(k-f[m-1])}{(f[m+1]-f[m-1])(f[m-f[m-1]]}}}{\frac{\frac{2(f[m+1]-k)}{(f[m+1]-f[m-1])(f[m-f[m-1]]}}{0}} \qquad Ec.\ (3.4)$$

Hay que tener en cuenta que la función $f[m]$ calcula los extremos en cada uno de los filtros triangulares siendo$f1\ y\ fh$ los puntos extremos inferiores y superiores.

Luego de multiplicar la señal de audio por los filtros se calculará la energía que corresponde a cada uno con la siguiente ecuación (3.5).

$$E_m = \sum_{k=0}^{N-1} |X[k]|^2 H_m[k] \qquad Ec.\ (3{,}5)$$

Luego de haber calculado la energía en cada filtro se debe calcular el logaritmo, para pasar al dominio de la potencia espectral logarítmica. Este cálculo genera una

dependencia entre los filtros y las bandas adyacentes que originan una correlación en los coeficientes espectrales la solución a esta correlación es aplicar la transformada cepstral o llamada también transformada directa del coseno, esta dependencia llevará los coeficientes al dominio del tiempo en la frecuencia transformándolos en coeficientes de MEL Cepstrales de un vector que se lo realizará con la siguiente ecuación (3.6) .

$$C_{MFCC}[m] = \sum_{k=0}^{N-1} \log(E_k) \cos\left(m\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right) \quad m = 1, \dots, F. \qquad \textit{Ec. (3.6).}$$

Teniendo en cuenta que cada comando de voz se lo grabará y procesará en diferentes ambientes tanto para la grabación de ejemplos y evaluación, el ruido, el tono de voz, las variaciones de los canales asignados a realizar, las adquisiciones de datos y las interrupciones mediante código son factores que afectarán el rendimiento del prototipo, aunque los vectores ya característicos de cada comando estén almacenados se los realizará cuatro veces por comando es decir existirán 16 vectores Cepstrales en total por los cuatro comandos pregrabados vectores que están anexados al final del documento. Culminado todo este proceso de obtención de coeficientes de MEL se procede ya a la inicialización del algoritmo, grabando los comandos de voz que se van a reconocer en el prototipo ensamblado en la figura (3.26) se observa la obtención de audio en presencia de ruido.

Figura 3. 26 Obtención de audio con Ruido

```
//sigue grabando pero cuando ya no hay ruido
while (i2s_rec_flag == 0)
    continue;
if (i2s_rec_flag == 1) {
    for (i = 0; i < frame_mov; i++)
        v_dat[i + frame_mov] = rx_buf[i];
} else {
    for (i = 0; i < frame_mov; i++)
        v_dat[i + frame_mov] = rx_buf[i + frame_mov];
}
i2s_rec_flag = 0;
while (1) {
    while (i2s_rec_flag == 0)
        continue;
    if (i2s_rec_flag == 1) {
        for (i = 0; i < frame_mov; i++) {
            v_dat[i] = v_dat[i + frame_mov];
            v_dat[i + frame_mov] = rx_buf[i];
        }
    } else {
        for (i = 0; i < frame_mov; i++) {
            v_dat[i] = v_dat[i + frame_mov];
            v_dat[i + frame_mov] = rx_buf[i + frame_mov];
        }
    }
    i2s_rec_flag = 0;
    if (VAD2(v_dat, valid_voice, &atap_arg) == 1)
        break;
    if (receive_char == 's') {
        *mtch_dis = dis_err;
```

Código de adquisición de comandos de voz para comparar con los audios ya pre grabados. Elaborado por: Jhon Cusco y John León

Ya adquiridos los datos necesarios y enviados por los arreglos declarados con anterioridad empieza el proceso de comparación de coeficientes entre los audios pregrabados y guardados en la tarjeta SD con los audios que el usuario está hablando frente al prototipo, cada ejemplo de audio está registrado con el mismo nombre pero con indicador de numero distinto identificados del 1 al 4, el listado de los comandos pregrabados que están listos para realizar su comparación se los representa de manera gráfica en la figura (3.27).

Figura 3. 27 Lista de comandos de voz pre grabados

```
digitalWrite(LED_RED, HIGH);
digitalWrite(LED_GREEN, HIGH);
digitalWrite(LED_BLUE, HIGH);

Serial.println("Initializing model...");

rec.addVoiceModel(0, 0, buscar_mesa_0, fram_num_buscar_mesa_0);
rec.addVoiceModel(0, 1, buscar_mesa_1, fram_num_buscar_mesa_1);
rec.addVoiceModel(0, 2, buscar_mesa_2, fram_num_buscar_mesa_2);
rec.addVoiceModel(0, 3, buscar_mesa_3, fram_num_buscar_mesa_3);
rec.addVoiceModel(1, 0, abrir_puerta_0, fram_num_abrir_puerta_0);
rec.addVoiceModel(1, 1, abrir_puerta_1, fram_num_abrir_puerta_1);
rec.addVoiceModel(1, 2, abrir_puerta_2, fram_num_abrir_puerta_2);
rec.addVoiceModel(1, 3, abrir_puerta_3, fram_num_abrir_puerta_3);
rec.addVoiceModel(2, 0, mover_silla_0, fram_num_mover_silla_0);
rec.addVoiceModel(2, 1, mover_silla_1, fram_num_mover_silla_1);
rec.addVoiceModel(2, 2, mover_silla_2, fram_num_mover_silla_2);
rec.addVoiceModel(2, 3, mover_silla_3, fram_num_mover_silla_3);
rec.addVoiceModel(3, 0, encender_equipo_0, fram_num_encender_equipo_0);
rec.addVoiceModel(3, 1, encender_equipo_1, fram_num_encender_equipo_1);
rec.addVoiceModel(3, 2, encender_equipo_2, fram_num_encender_equipo_2);
rec.addVoiceModel(3, 3, encender_equipo_3, fram_num_encender_equipo_3);
```

Comparación de archivos grabados y audios pregrabados por código de programación. Elaborado por: Jhon Cusco y John León.

3.5 IMPLEMENTACIÓN DEL PROTOTIPO

El prototipo es adaptable ya que todas sus piezas se las adquiere por separado, existen dos modelos en el mercado uno de 90x 61.5x9.5mm y el otro de 3.54x2.42x0.37 respectivamente, nuestro prototipo es el primero ya que se necesita que sea transportable y que funcione con batería. A continuación, se muestran las imágenes del prototipo en la figura (3.28)

Figura 3. 28 Prototipo MAIX GO

El prototipo MAIX GO de SIPEED con todas sus partes sin ensamblar y los componentes adquiridos para implementar el algoritmo. (ROBOT, 2018)

Figura 3. 29 Prototipo MAIX GO ensamblado

El prototipo MAIX GO de SIPEED con todas sus partes ya ensambladas para uso sin cables de conexión ni fuente externa. Elaborado por: Jhon Cusco y John León

CAPÍTULO 4
PRUEBAS Y RESULTADOS

4.1 PRUEBAS DE LECTURA DE DATOS CON EL MICRÓFONO

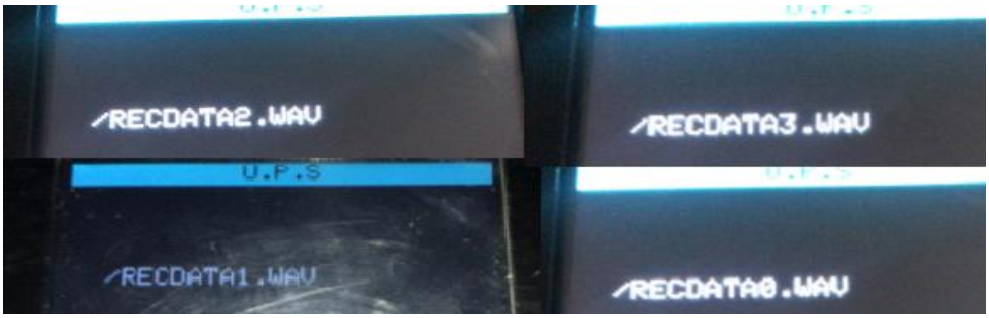
Las primeras pruebas con el micrófono del prototipo se fueron realizadas con la ayuda de un computador para lograr verificar mediante comunicación serial si el prototipo está o no captando datos, esto con la finalidad de familiarizarse con el funcionamiento del módulo, ya que Visual Studio permite revisar paso a paso el estado de los datos recibidos y enviados al microcontrolador, se los representa de manera gráfica en la figura (4.1)

Figura 4. 1 Terminal de conexión serial de Visual Studio

Visualización de los comandos enviados a través de comunicación serial desde el prototipo hacia el PC

Las primeras pruebas realizadas fueron con el primer código de programación que es el encargado de almacenar los comandos de voz, que servirán como respuesta correcta luego de procesar las señales de audio habladas por el individuo respuestas que se pueden representar de manera gráfica en la figura (4.2).

Figura 4. 2 Pantalla LCD MAIX GO



Las fotografías muestran los nombres con los que guarda los audios que el usuario a grabado como respuesta al algoritmo

Para realizar estas pruebas se utiliza el código explicado en el apartado 3.4 elaborado en PLATFORMIO con código de Arduino. Se enviaron 4 comandos de voz que están ordenados en la tabla (4.1).

Tabla 0.1. Prueba de almacenamiento de respuestas a comandos de voz.

Palabras enviadas	**Datos de audio almacenados**
ABRIR PUERTA	4
BUSCAR MESA	4
MOVER SILLA	4
ENCENDER EQUIPO	4
Promedio de error en conexión	**0%**

Pruebas de almacenamiento de datos en tarjeta SD del prototipo Elaborado por: Jhon Cusco y John León

En la tabla 4.1 se observa que el porcentaje de error en la conexión serial y almacenamiento de comandos es nulo y que se puede confiar en que el primer código implementado al prototipo funciona con la alta fiabilidad a la palabra hablada por el usuario.

4.2 PRUEBAS DE CONEXIÓN Y ENVÍO DE DATOS A LA PC

Para seguir con las pruebas de funcionamiento se realizó el proceso de compilación y carga del nuevo archivo programado, se utiliza el código explicado en la sección 3.4.1, se enviaron 4 comandos de voz al microcontrolador del prototipo conectado a través de un cable tipo C a una computadora portátil para verificar si los datos están siendo almacenados en la tarjeta SD, como era de esperarse, si ya en la prueba de lectura de datos fue correcta en esta de igual manera se verifico el almacenamiento y l carga de datos a la tarjeta a través del programa VISUAL STUDIO. Los resultados se los representa de manera detallada en la tabla (4.2).

Tabla 0.2. Prueba de conexión entre el Prototipo y la PC

Palabras enviadas	**Datos de audio por comunicación serial**
ABRIR PUERTA	4
BUSCAR MESA	4
MOVER SILLA	4
ENCENDER EQUIPO	4
Promedio de error en conexión	**0%**

Pruebas de envío de datos mediante conexión serial hacia la PC. Elaborado por: Jhon Cusco y John León

En la tabla 4.2 se observa que el porcentaje de error en el envío de datos serial entre el Prototipo y la PC es también nulo y que se determina que el programa que se usa en el computador obtiene una comunicación serial de manera correcta y escribirá la palabra acorde a la hablada y responderá con alta fiabilidad, asegurando que los vectores pregrabados son funcionales y se almacenaran en la tarjeta SD para su evaluación con los comandos de voz dictados por el usuario.

4.3 PRUEBAS PARA RECONOCER PALABRAS DE VOZ

Tabla 0.3. Prueba de reconocimiento de palabras clasificadas

Palabras usadas	**Datos reconocidos de 20**	
	Usuario 1	**Usuario 2**
ABRIR PUERTA	20	20
BUSCAR MESA	20	19
MOVER SILLA	19	20
ENCENDER EQUIPO	19	19
PORCENTAJE DE ERROR	**2.5%**	**2.5%**
PORCENTAJE DE FUNCIONAMIENTO	**97.5 %**	**97.5 %**

Pruebas de reconocimiento de palabras. Elaborado por: Jhon Cusco y John León

Por otro lado, dentro de las mismas pruebas realizadas se hicieron ya las de reconocimiento de voz, realizadas por dos personas con diferentes tonos de voz, y se verifica que el porcentaje de error es casi nulo ya que todas las palabras fueron procesadas correctamente y reconocidas de manera exitosa a no ser por un par de ocasiones en los que el sonido demasiado alto del ruido externo afecta la percepción de comandos de voz que el usuario está solicitando sean reconocidos por el prototipo. Estos resultados fueron logrados gracias a que por cada comando de voz se grabaron 4 ejemplos por cada uno, al igual que se lo realiza por cada persona que quiera usar el prototipo para reconocer dichos comandos con su propia voz.

CONCLUSIONES

- De acuerdo con el primer objetivo que fue analizar el estado del arte de sistemas de reconocimiento de voz, se concluye que la tarea de reconocimiento no es tarea fácil, ya que depende de varios de factores, como las condiciones del ambiente sin ruido, las técnicas que se utilicen para la captura y filtrado de la señal de audio, los métodos que se usen para el tratamiento de datos y para el reconocimiento de palabras, son tareas que inmiscuyen un alto nivel de cálculo computacional y requieren de computadoras con procesadores que puedan solventar estas necesidades sin llegar a un desborde de sus microprocesadores y por ende perdida de información y un mal funcionamiento del equipo. Por todo lo mencionado anteriormente se trató de evitar el usar métodos que requieran un alto nivel de cálculo y computadoras con procesadores de buenas prestaciones en el presente trabajo.

- Al cumplir otros de los objetivos se logra verificar que si se captura audio sin un previo procesamiento con filtros como los ya establecidos por los MFCC no se lograra evitar el ruido que el micrófono escucha y captara señales erróneas, que se debieron eliminar utilizando una condición de intervalo de reconocimiento para procesar datos tal como se lo realizo en el desarrollo de la programación con la librería maix_speed_recognition que elimina automáticamente el ruido por las escalas logarítmicas que el prototipo incluye.

- En las pruebas realizadas en el reconocimiento de palabras con el prototipo presentado en este trabajo se obtuvo un porcentaje de identificación de 97.5%, resultados que fueron obtenidos gracias a que el prototipo cuenta con un microcontrolador de gama alta con varios aceleradores y periféricos que son los más efectivos al momento de trabajar con audio.

RECOMENDACIONES

- Se recomienda probar el prototipo en un ambiente controlado y libre de ruido para no afectar al sistema y evalúe datos erróneos. Además, añadido a esto, se debe instruir al usuario para que sepa cómo utilizar el prototipo y sus diferentes funciones.

- Es preciso tener mucho cuidado al momento de alimentar el sistema ya que la fuente de alimentación no debe sobrepasar los 5 voltios para no ocasionar daños. Para evitar esto, si en alguna ocasión es necesario cambiar la batería del prototipo se recomienda hacerlo por una que tenga una salida de 5 v

- Se bebe tener precaución de que el micrófono este a una distancia considerable a la que no afecte la respiración del usuario en la adquisición de datos del sistema. Para esto es recomendable colocarlo a 5 centímetros de la boca ya que a esa distancia fue probado.

REFERENCIAS BIBLIOGRÁFICAS

Mermelstein , P., & S, D. (Agosto 1980). *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences.* Santa Barbara, CA: IEEE Transactions on Acoustics, Speech, and Signal Processing (Volume: 28 , Issue: 4.

Amazon-MAX9814. (2020). *Módulo de amplificador de micrófono Electret + control de ganancia automático AGC de 5 pines 2,7 V-5,5 V 3 mA*. Obtenido de https://www.amazon.es/amplificador-micr%C3%B3fono-Electret-ganancia-autom%C3%A1tico/dp/B07SWBZRSR

Aprendiendo_Arduino. (2018). *Módulo micrófono Arduino*. Obtenido de https://aprendiendoarduino.wordpress.com/2018/10/16/modulo-microfono-arduino/

Asnawi, R., & Said , M. (2018). Testing of Other Languages Usage in Addition to the Default Languages for the. *2018 International Conference on Electronics Technology*, 4.

Betancourt, H., Armijos, D., Ponce, A., & Ortega , F. (2018). Portable Expert System to Voice and Speech. *2018 2nd European Conference on Electrical Engineering and Computer Science (EECS)*, 5.

Cruz, A. (15 de Junio de 2017). Obtenido de Reconocimiento de Voz usando Redes Neuronales Artificiales Backpropagation y Coeficientes LPC: file:///C:/Users/USER/Downloads/ReconocimientodeVozusandoRedesNeuronalesArtificialesBackpropagationyCoeficientesLPCLuisBeltranAcevedo%20(1).pdf

De Luna, M. M. (Diciembre de 2006). *Conciencia tecnilogica*. Obtenido de Reconocimiento de Voz con Redes Neuronales, DTW y Modelos Ocultos de: https://www.redalyc.org/pdf/944/94403203.pdf

DFROBOT. (2019). *DFPlayer mini* . Obtenido de https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

Fernandez, J. (21 de febrero de 2019). *Control de Arduino por reconocimiento de voz* . Obtenido de https://lacienciaparatodos.wordpress.com/2019/02/21/control-de-arduino-por-reconocimiento-de-voz/

García Serrano, A. (2013). Inteligencia artificial. Fundamentos, práctica y aplicaciones. México: Alfaomega Grupo Editor S.A.

García, S. (2019). *Tipos de microfonos según su construcción*. Obtenido de Manual para radilistas analfatécnicos: https://www.analfatecnicos.net/pregunta.php?id=35

González, J. R. (Septiembre de 2019). Obtenido de Evaluación comparativa de sistemas de reconocimiento de locutor basados en los algoritmos LPC, CC y MFCC: http://revistas.um.edu.uy/index.php/ingenieria/article/view/390/480

Kamdar, B., MIrchandani, B., & Shah, D. (2012). Real Time Speech Recognition using IIR Digital Filters Implemented on an Embedded System. *2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India*, 5.

MAIXDUINO. (15 de MAYO de 2019). *SIPEED*. Obtenido de SIPEED: https://github.com/sipeed/Maixduino/blob/master/libraries/Maix_Speech_Recognition/src/util/MFCC.c

Martínez, A. (18 de Noviembre de 2013). *Ingenius*. Obtenido de Reconocimiento de voz basado en MFCC, SBC y Espectrogramas: file:///C:/Users/USER/Downloads/351-Texto%20del%20art%C3%ADculo-1166-1-10-20160118.pdf

MathWorks, R. N. (enero de 2018). *Redes neuronales.* Obtenido de ¿Qué es una red neuronal?: https://la.mathworks.com/discovery/neural-network.html

MAX9814. (2020). *Micrófono Amplificador*. Obtenido de https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf

Microchip. (2019). *PIC16F87X Data Sheet*. Obtenido de https://www.alldatasheet.com/view.jsp?Searchword=Pic16f877&gclid=EAIaIQobChMIwaeynInz5gIVCZyzCh0S5Ag8EAAYAiAAEgJDJPD_BwE

MikroElektronika. (2020). *MIKROE*. Obtenido de MikroElektronika Books-Introducción al mundo de los microcontroladores: https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/introduccion-al-mundo-de-los-microcontroladores

Oropeza Rodríguez, J., & Suárez Guerra, S. (2006). Algoritmos y métodos para el reconocimiento de Voz en Español Mediante Sílabas. *Computación y Sistemas Vol. 9*, 270-286.

Oropeza, J. (15 de Diciembre de 2006). *Scielo*. Obtenido de Algoritmos y Métodos para el Reconocimiento de Voz en Español Mediante Sílabas: http://www.scielo.org.mx/scielo.php?pid=S1405-55462006000100007&script=sci_arttext

Raczynski, M. (21 de Junio de 2018). *IEEE XPLORE*. Obtenido de Speech processing algorithm for isolated words recognition: https://ieeexplore.ieee.org/abstract/document/8388238

Raczynski, M. (2018). Speech processing algorithm for isolated words recognition. *2018 IEEE*, 5.

ROBOT, D. (26 de 12 de 2018). *DF ROBOT* . Obtenido de https://www.dfrobot.com/product-1975.html

Robótica, R.-A. y. (2020). *Módulo para reconocimiento de voz y micrófono Arduino*. Obtenido de https://rambal.com/audio-sonido/1061-modulo-elechouse-reconocimiento-de-voz-y-microfono-arduino.html

Rojo, L. (Abril de 2011). Obtenido de MEJORAS EN RECONOCIMIENTO DEL HABLA: http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20110603LeticiaRueda.pdf

Rueda, L. (Abril de 2011). *MEJORAS EN RECONOCIMIENTO DEL HABLA*. Obtenido de MEJORAS EN RECONOCIMIENTO DEL HABLA: file:///C:/Users/USER/Downloads/20110603LeticiaRueda%20(1).pdf

sad. (dsad). *asd*. dasdsa: ads.

Thiang. (2007). Implementation of Speech Recognition on MCS51 Microcontroller for Controling Wheelchair. *International Conference on Intelligent and Advanced Systems 2007*, 6.

Tiang. (Noviembre de 2007). Obtenido de Implementation of Speech Recognition on MCS51 Microcontroller for Controlling Wheelchair: https://www.researchgate.net/publication/43649432_Implementation_of_Speech_Recognition_on_MCS51_Microcontroller_for_Controlling_Wheelchair

Urrere, L. (8 de septiembre de 2016). *Redes Neuronales MATLAB*. Obtenido de https://www.youtube.com/watch?v=Nw17-YBCQ_E