UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

SISTEMA DOMÓTICO BASADO EN LA PLATAFORMA RASPBERRY PI Y COMUNICACIONES INALÁMBRICAS CONTROLADO MEDIANTE ÓRDENES DE VOZ EN UN DISPOSITIVO ANDROID

TESIS
PARA OBTENER EL GRADO DE:

MAESTRO EN ELECTRÓNICA
OPCIÓN: SISTEMAS INTELIGENTES APLICADOS

PRESENTA:

ING. JESÚS SAMUEL ORTIZ IBARRA

DIRECTOR DE TESIS:

DR. ROSEBET MIRANDA LUNA

CODIRECTOR DE TESIS:

DR. JOSÉ ANÍBAL ARIAS AGUILAR

H. CD. DE HUAJUAPAN DE LEÓN, OAXACA; ENERO DE 2019

Dedicatoria

Este trabajo de tesis está dedicado a la memoria de mi abuelita Mari, mi abuelito Chano y mi tía Yeya, con quienes me hubiera gustado compartir este logro.

A mi padre y a mi madre, por su atención y cuidado. Este trabajo es el resultado de su esfuerzo.

A mis hermanos, ya que gracias a ellos he aprendido cosas que nunca hubiera podido aprender solo, su compañía ha sido esencial en este viaje.

Jesús Samuel.

Agradecimientos

A los directores de este trabajo de tesis, por su tiempo y su paciencia.

A Heriberto, por sus consejos y su amistad.

Jesús Samuel.

Índice General

Índice de Figuras

Índice de Tablas

Resumen

En la actualidad, el reconocimiento de voz es una de las áreas del conocimiento con mayor énfasis debido a la disponibilidad de una gran cantidad de dispositivos con capacidad de procesamiento para realizar tareas relacionadas a dicho reconocimiento. Por otro lado, en el área de la domótica, los avances en tecnología inalámbrica permiten desarrollar sistemas cada vez más sofisticados. Este trabajo presenta una investigación en ambas áreas con la finalidad de desarrollar un sistema domótico que funcione mediante órdenes de voz.

Se propone un sistema basado en una aplicación para dispositivo móvil con sistema operativo Android y un sistema de control en una tarjeta de desarrollo Raspberry Pi. Este sistema tiene una aplicación, instalada en un dispositivo móvil, que recibe órdenes por voz del usuario y las decodifica usando un vocabulario limitado que no requiere de una conexión a internet. El dispositivo móvil envía la información codificada mediante Wi-Fi a un sistema de control, implementado en una tarjeta de desarrollo Raspberry Pi, para que finalmente, dicho sistema de control ejecute las órdenes del usuario a través de los actuadores correspondientes.

Se obtuvo un sistema funcional, el cual puede ser usado por personas con movilidad corporal restringida y que no pueden utilizar sistemas domóticos convencionales. De esta manera se espera brindar autonomía en la realización de algunas tareas comunes.

Capítulo 1. Introducción

1.1. Estado del Arte

El reconocimiento de voz hace referencia a la implementación de un programa que pueda interpretar y reconocer las palabras u órdenes emitidas mediante voz para realizar una tarea requerida. Todo esto mediante el uso de una computadora, la cual se encarga de identificar los patrones de voz similares y, con base en ello, tomar una decisión acerca del resultado que se espera (Saini y Kaur, 2013).

Recientemente, el reconocimiento de voz es uno de los temas de investigación que ha tomado un auge sin precedente gracias a la miniaturización de la tecnología y al incremento en capacidad de procesamiento y de interconectividad de dispositivos móviles. Ejemplos de estos sistemas son Siri de Apple, Ok Google de Google y Cortana de Microsoft, que mejoran la interactividad entre los usuarios y las plataformas en los cuales están disponibles, al realizar tareas como búsqueda de contenidos y realizar acciones dentro del sistema operativo, entre otras (Huang, Baker y Reddy, 2014).

Cabe mencionar que estos sistemas son suficientemente complejos para realizar las tareas que les son requeridas, por ello, requieren de una infraestructura robusta respecto a recursos informáticos y de hardware, para procesar los datos de un gran número de usuarios de manera eficaz y concurrentemente. Dentro de estos sistemas existen subsistemas más compactos y mucho menos sofisticados como los descritos por Dhingra, Nijhawan y Pandit (2013), Mohan y Ramesh (2014), De Luna Ortega, Martínez Romo y Mora González (2006) y Darabkh, Khalifeh, Bathech y Sabah (2013), los cuales ayudan en tareas específicas como interpretar órdenes y palabras específicas usando un vocabulario restringido (Shaikh, Mesquita y Das Chagas, 2017).

Por otra parte, el estudio de la domótica es un campo que considera diversas soluciones, éstas van desde sistemas de seguridad, manejo de electrodomésticos y automatización de tareas del hogar (Ghazal y Al-Khatib, 2015). La creciente integración del internet de las cosas (IoT por sus siglas en inglés) hace posible que el desarrollo de estos sistemas sea cada vez más complejo e integral, al mismo tiempo que brindan al usuario mayor independencia para realizar tareas de manera fácil y organizada (Abd Wahab, 2016).

Los sistemas automatizados por control de voz presentan una ventaja al usuario cuando se relacionan con sistemas que requieran métodos para acceder fácilmente a todas sus funcionalidades. Este tipo de sistemas implementan comúnmente para asistencia de personas que poseen algún tipo de discapacidad y, en algunos casos, para personas de la tercera edad que no pueden ser completamente autosuficientes, tal como se describe en los estudios que presentan Nishimori, Saitoh y Konishi (2007), Rojas Rodríguez y otros (2015), Abd Wahab (2016), Lv y Zhang (2008) y Chen, Wu, Chen, Chin y Chen (2017).

Por otro lado, existen sistemas domóticos que utilizan redes neuronales para aprender el comportamiento de sus usuarios y que se adapte a éstos (Hussein, Mehdi, Mirna y Fahs, 2014), y otros utilizan hardware especializado para el reconocimiento de voz junto con un microcontrolador para llevar a cabo dicha automatización (Mittal y otros, 2015), (Prabhu y Pradeep, 2016).

En 2013, Oltenau, Oprina, Tapus y Zeisberg propusieron un sistema de automatización de una casa basado en un dispositivo con sistema operativo Android conectado mediante los protocolos de comunicaciones ZigBee, Internet y Wi-Fi (véase Figura 1.1). En este trabajo utilizan una computadora como puerta de enlace (*Gateway*) con los dispositivos e implementaron la comunicación mediante un adaptador (*USB Dongle*) que se inserta en el puerto USB del dispositivo móvil. Para el control de los dispositivos ZigBee se emplea una interfaz con la que los usuarios deciden las acciones a realizar con cada uno de los dispositivos conectados a la red.

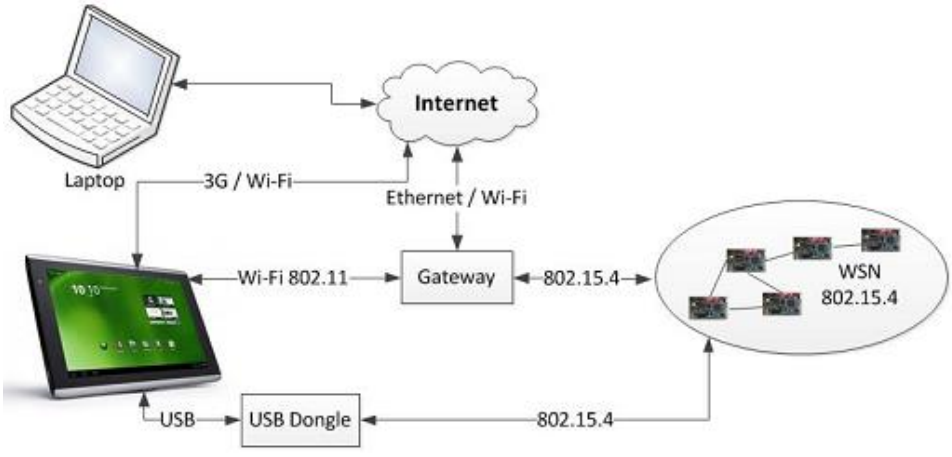


Figura 1.1. Sistema desarrollado por Oltenau et al. (2013).

Ese mismo año, Baraka, Ghobril, Malek, Kanj y Kayssi (2013) presentaron un sistema de automatización de una casa usando una aplicación diseñada para dispositivos con sistema operativo Android, la cual se conectaba mediante Internet a una puerta de enlace implementada en una tarjeta de desarrollo Arduino (véase Figura 1.2). Esta puerta de enlace se comunica con dispositivos ZigBee y X10, generando un sistema híbrido para su control. Se utiliza una interfaz gráfica de usuario (GUI), con la cual los usuarios pueden programar tareas y controlar los dispositivos disponibles en el sistema.

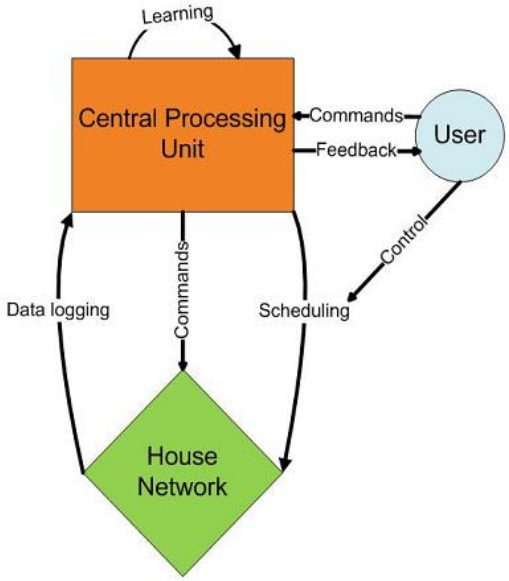


Figura 1.2. Diagrama del sistema diseñado por Baraka et al. (2013).

En 2015, Camargo, Coronel y Calderón diseñaron un sistema de automatización de una casa basado en Google Voice API y en un conjunto de redes neuronales que hacen uso de *feedforward-backpropagation* para imitar una conversación humana y ejecutar órdenes por voz de una manera amena. En ese trabajo se usa una tarjeta de desarrollo Arduino Mega ADK que implementa un Host USB para proporcionar comunicación entre la aplicación en Android y la tarjeta de desarrollo. Cabe señalar que no se mencionan los protocolos de comunicación ni los dispositivos que se utilizaron en la realización de este trabajo.

1.2. Planteamiento del Problema

El presente trabajo de investigación propone la implementación de un sistema de reconocimiento de voz para el control automatizado de una casa, que con base en la emisión de una orden (*utter*) se ejecute una acción específica.

Se implementará un sistema domótico monousuario de fácil uso, de tal manera que si los usuarios tienen movilidad corporal reducida éstos puedan hacer uso del sistema. Con éste se pretenden controlar dispositivos como lámparas y ventiladores colocados en diferentes puntos de la casa, para lograr esto se propone el sistema planteado en la Figura 1.3.

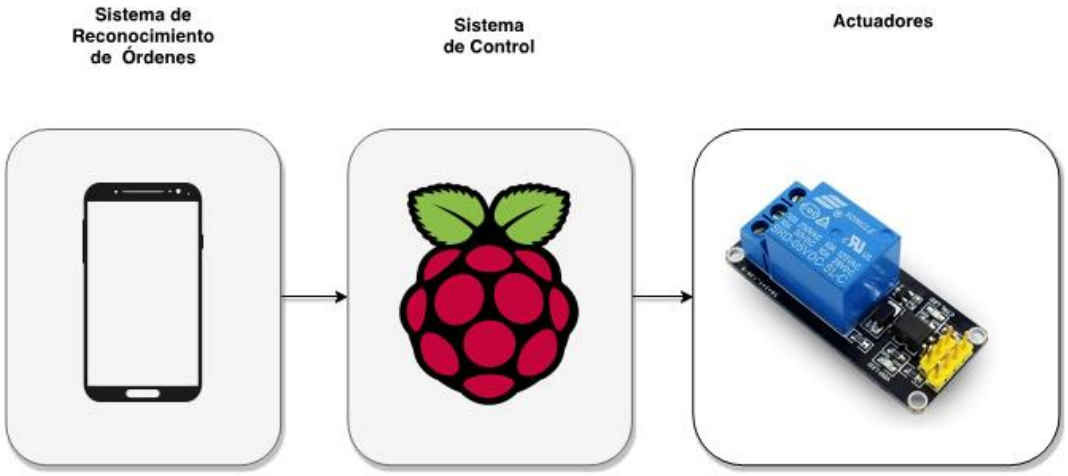


Figura 1.3. Diagrama a bloques del sistema.

En este sistema se hará uso de un dispositivo móvil, con sistema operativo Android, para captar y hacer el reconocimiento de órdenes.

El sistema de control se implementará en una plataforma Raspberry Pi, la cual esperará las órdenes mediante Wi-Fi, después de recibirlas las decodificará para finalmente enviarlas a los actuadores correspondientes; las señales serán enviadas a los actuadores mediante Wi-Fi o Bluetooth según sea el caso. El sistema de reconocimiento de órdenes estará integrado como muestra la Figura 1.4, mientras que el sistema de control contará con las fases que se muestran en la Figura 1.5. En caso de que uno de los actuadores llegase a fallar, los demás actuadores seguirían funcionando.

Figura 1.4. Diagrama de sistema de reconocimiento de órdenes.

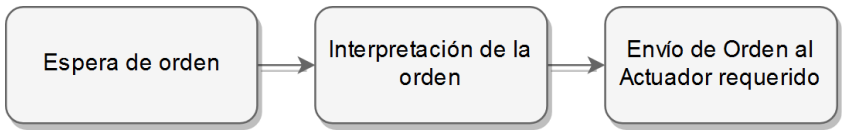


Figura 1.5. Diagrama del sistema de control.

1.3. Justificación

El diseño e implementación de un sistema de automatización para el control de una casa puede ayudar a los usuarios a mejorar su calidad de vida, no obstante, los sistemas hasta hoy desarrollados se basan en interfaces cuyos medios de entrada requieren de cierta movilidad en alguna parte del cuerpo, por tanto, desestiman a aquellos usuarios que no cuentan con dicha posibilidad o que están restringidos por su condición. Por ejemplo, el dispositivo McTin usado por Chen, Wu, Chen, Chin y Chen (2017) hace uso de un dispositivo bucal y la interfaz gráfica de usuario, diseñada por Rojas-Rodríguez *et al.* (2015), requiere que se toque la pantalla para activar los dispositivos en casa.

En México, la prevalencia de la discapacidad para el 2014 fue del 6% (ENADID, 2014), que representa un 7.1 millones de habitantes que no pueden o tienen mucha dificultad para hacer alguna de sus actividades cotidianas, de esta población aproximadamente la mitad son adultos mayores. Cabe señalar que Oaxaca posee el 3.6% de la población total de discapacitados (Instituto Nacional de Estadística y Geografía, 2017).

Respecto a las personas de la tercera edad, el Censo de Población y Vivienda 2010 reportó una población de 10,055,379, que significa un 8.96% de la población total. Y en Oaxaca habita un 10.7% de adultos mayores (Instituto Nacional de las Personas Adultas Mayores, 2016) muchos de los cuales podrían beneficiarse del sistema propuesto.

En este trabajo se propone un sistema domótico controlado por órdenes de voz que ayudará a sus usuarios en mejorar su calidad de vida y por ende su bienestar en general.

1.4. Hipótesis

Es posible diseñar e implementar un sistema domótico basado en una tarjeta de desarrollo Raspberry Pi controlada inalámbricamente mediante órdenes de voz desde un dispositivo móvil con sistema operativo Android.

1.5. Objetivos

1.5.1. Objetivo general

Diseñar e implementar un sistema domótico basado en una tarjeta de desarrollo Raspberry Pi controlada inalámbricamente mediante órdenes de voz desde un dispositivo móvil con sistema operativo Android.

1.5.2. Objetivos específicos

Con la intención de cumplir con el objetivo general se plantean los siguientes objetivos específicos:

- Diseñar e implementar un sistema de reconocimiento de voz en un dispositivo móvil con sistema operativo Android.
- Diseñar e implementar un sistema de comunicación para el sistema domótico.
- Implementar un sistema de actuadores.
- Implementar el sistema de control mediante órdenes de voz.
- Evaluar el rendimiento del sistema.

1.6. Metas

El desarrollo de este trabajo plantea cumplir con las siguientes metas:

- Implementación del módulo para la activación del sistema de reconocimiento de voz.
- Diseño del vocabulario del sistema de reconocimiento de voz.
- Adquisición y etiquetado de un conjunto de entrenamiento con las palabras que vayan a ser utilizadas como órdenes para obtener una mejor respuesta.
- Diseño e implementación en una PC de un sistema de reconocimiento de voz monolocutor cuyo vocabulario incluya las órdenes y modificadores necesarios.
- Entrenamiento del sistema de reconocimiento de voz para que interprete las órdenes emitidas por el usuario.
- Diseño e implementación de un sistema de validación de órdenes para el correcto funcionamiento del sistema.

- Implementación del sistema de reconocimiento en un dispositivo móvil con sistema operativo Android.
- Entrenamiento de la implementación para que funcione en el dispositivo móvil, cumpliendo con reglas establecidas.
- Realización de pruebas del sistema de reconocimiento de voz.
- Diseño e implementación del sistema de comunicación Wi-Fi para comunicar el dispositivo Android y la plataforma Raspberry Pi.
- Realización de pruebas del sistema de comunicación Wi-Fi.
- Diseño e implementación del sistema de comunicación inalámbrico para comunicar la plataforma Raspberry Pi y los actuadores.
- Pruebas del sistema de comunicación inalámbrico de los actuadores.
- Configuración de los actuadores a la plataforma Raspberry Pi.
- Implementación de un intérprete de las órdenes emitidas por el dispositivo móvil en la plataforma.
- Realización de pruebas de caja gris del sistema domótico.
- Realización de pruebas de caja negra del sistema domótico.

1.7. Limitaciones

Una investigación conlleva limitaciones que son dadas por los medios materiales y por el tiempo disponible para realizar la investigación. Por lo tanto, es prudente delimitar el alcance de este trabajo considerando lo siguiente:

- El sistema realizará un preprocesamiento general para todos los datos de entrada sin importar su longitud.
- El sistema tendrá un vocabulario limitado, menor de 100 palabras.
- Se despreciará el tiempo de procesamiento que ocupe el dispositivo móvil en detectar la orden debido a que este proceso utiliza varias operaciones matemáticas.
- El número de dispositivos a controlar será reducido a 6 actuadores debido a limitaciones económicas.
- Los usuarios harán las pruebas desde una sola posición, en este caso debido a la falta de personas que presenten condiciones de movilidad reducidas.

1.8. Metodología de Desarrollo

La metodología de desarrollo considera una modificación a la propuesta por Arnold Berger (2002), la cual propone las siguientes fases: especificación de producto, partición del Hardware y Software, iteración e implementación, diseño detallado del Hardware y Software, integración, pruebas iterativas, liberación del producto y mantenimiento. Sin embargo, en este trabajo es necesaria una fase para el desarrollo de la interfaz de usuario por voz, que considere examinar los patrones lingüísticos y de comportamiento del usuario respecto al sistema (Cohen , Giangola y Balogh, 2004), para ello, es necesario que en esta fase se realicen los procesos de selección de vocabulario, selección de comportamientos y predefinición de acciones, los cuales serán el soporte principal de lo que se implementará en hardware.

La fase de desarrollo de interfaces de usuario por voz proporcionará la especificación del software y de vocabulario, y será el punto de partida para el diseño del hardware, debido a que el diseño de la interfaz por voz (VUI, por sus siglas en inglés) lleva a cabo un estudio más profundo en los requerimientos del sistema. Por lo tanto, el desarrollo de Hardware y de Software no será un proceso simultáneo sino un proceso que requerirá como punto de partida el desarrollo de la interfaz de usuario por voz y la especificación del vocabulario. El proceso de diseño e implementación de Hardware considera fases de pruebas e integración de manera cíclica hasta que el sistema responda adecuadamente. En el Figura 1.6 se ilustra la metodología propuesta.

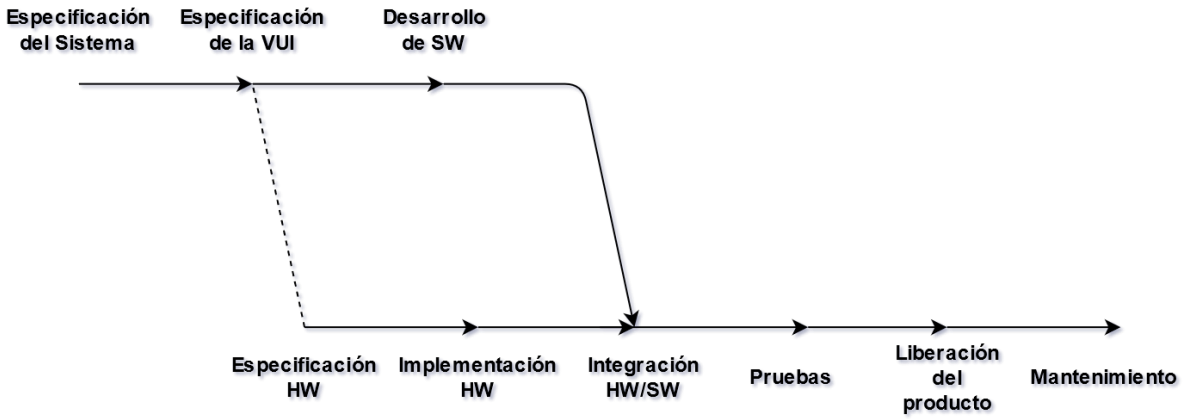


Figura 1.6. Diagrama de la metodología de desarrollo.

Capítulo 2. Marco Teórico

2.1. Reconocimiento de Voz

El objetivo del reconocimiento de voz es identificar, mediante el uso de algoritmos implementados en una computadora, las combinaciones de palabras que emite un usuario para conocer qué es lo que éste trata de comunicar. Para cumplir con el objetivo se pueden realizar las siguientes tareas: a) convertir voz en texto, b) procesar palabras con significado especial para ejecutar órdenes hacia algún sistema externo, o c) la creación de modelos que emulen la comunicación verbal, ya que como mencionan Anusuya y Katti (2009) el habla es la manera más natural de comunicación entre humanos.

Rabiner y Biing-Hwang (1993) mencionan que la realización de las tareas anteriores presenta un problema complejo y multidisciplinario, que abarca: a) procesamiento de señales, para extraer la información relevante de la señal de habla (*speech signal*) de una manera robusta y eficiente; b) acústica, encargada de entender la relación entre el habla física y el mecanismo fisiológico; c) reconocimiento de patrones, para crear datos de los patrones del habla; d) lingüística, que examina la relación entre sonidos y sintaxis; e) fisiología, para entender los mecanismos humanos del sistema nervioso encargados de la producción y percepción del habla; f) ciencias de la computación, para estudiar los algoritmos más eficientes; y g) psicología, encargada de permitir que la tecnología sea usada por humanos para resolver tareas prácticas.

No todos los sistemas de reconocimiento de voz tienen la misma tarea a resolver, por ello se clasifican de acuerdo con diversas características, las principales son: a) identificación de fonemas, b) identificación de palabras completas, y c) tamaño del vocabulario. Respecto al tamaño del vocabulario, Brown (2005) clasifica los sistemas de reconocimiento de voz en:

- Vocabularios pequeños (10-100 palabras).
- Vocabularios medianos (100-1000 palabras).
- Vocabularios grandes (más de mil palabras).
- Vocabularios muy grandes (es similar al anterior, pero se incluye el diálogo multimodal y el diálogo hablado).

Aparte de esta clasificación, los sistemas de reconocimiento de voz se dividen por el tipo de reconocimiento que realizan, Holmes y Holmes (2001) identifican tres tipos de sistemas:

- Sistemas de reconocimiento de palabras aisladas: detectan la enunciación de una palabra en específico dentro del vocabulario establecido.
- Sistemas de reconocimiento con palabras conectadas: detectan, dentro de un audio, la combinación de palabras con mejor calificación (*score*), algunas veces se hace uso de ciertas normas sintácticas definidas por el diseñador del sistema, no obstante, esto último no es necesario. Así mismo, para la detección de estas combinaciones se considera un número finito de palabras reconocidas, y un punto de inicio y uno de finalización de la enunciación.
- Sistemas de reconocimiento continuo: estos sistemas forman combinaciones de palabras mediante la detección de palabras conforme se enuncian; para esto se desconoce el número de palabras a reconocer y, por lo tanto, se carece de un final específico de palabras.

Aunado a lo anterior, también se considera al usuario final, y para esto Xuedong (1993) realiza la siguiente clasificación de sistemas de reconocimiento de voz:

- Dependientes del usuario: se consideran productos rígidos debido a que su uso está enfocado en un sólo usuario.
- Independientes del usuario: están enfocados a ser utilizados por múltiples usuarios, por ello, consideran características esenciales de la producción de voz y eliminan aquellas que son irrelevantes.
- Adaptivos: se adecuan al usuario final mediante la utilización de características de los sistemas independientes del usuario para adaptarlos a sistemas dependientes del usuario.

En algunos otros casos, el propósito es reconocer al usuario que está hablando con base en las características especiales de su voz; para esto se requiere de diferentes fases y procesamientos de la señal de voz. La Figura 2.1 muestra, de manera general, las fases de un sistema de reconocimiento de voz, las cuales se describen a continuación.

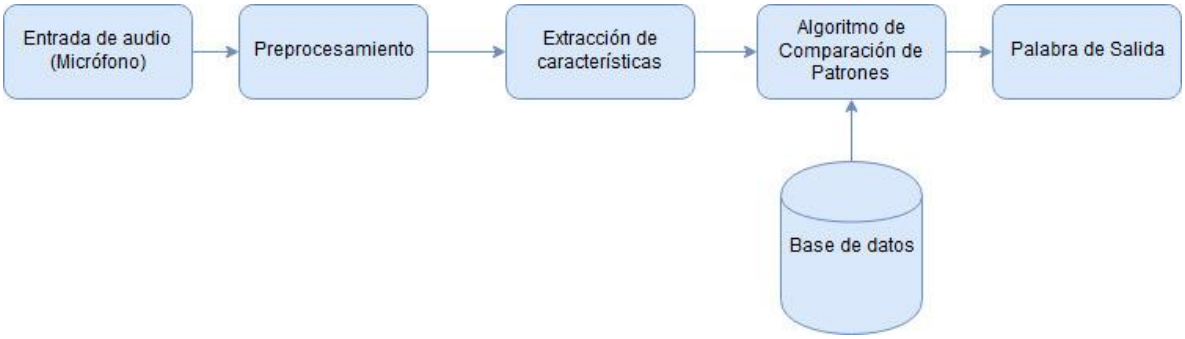


Figura 2.1. Fases de un sistema de reconocimiento de voz.

2.1.1. Preprocesamiento

El preprocesamiento de las señales de audio es una de las fases iniciales en un sistema de reconocimiento de voz. En esta fase se modifican las señales de audio obtenidas mediante los micrófonos para acentuar determinadas características de las señales de voz. Todo esto con el propósito de mejorar el rendimiento de dichos sistemas, ya que éste depende directamente de la calidad de la señal de habla, para lograr este propósito la señal original es segmentada tal como se menciona en el apartado dedicado a esta operación ya que no solo es utilizada en éste, sino también es utilizada en el proceso de extracción de características. Como se menciona este preprocesamiento se usa para eliminar datos que son irrelevantes para el proceso de reconocimiento de voz tales como el estado de ánimo y características acústicas del ambiente en que se encuentra el sistema (Kolokolov, 2003).

Esta fase considera dos procesos: preénfasis y detección de actividad de voz (VAD, por sus siglas en inglés), este último consiste en la detección de inicio y fin de presencia de voz, así como de silencios entre pronunciaciones. A continuación, se describen ambos procesos.

2.1.1.1. Preénfasis

Antes de llevar a cabo cualquier proceso de extracción de características se necesita realizar un filtrado mediante una ecuación de diferencias, ésta se aplica para todas las muestras de audio que están en la señal $s\{n = 1, N\}$ donde N indica el número de muestras de audio. A este proceso se le llama preénfasis (Young et al., 2017) y loa muestra la ecuación (1).

$$s'(n) = s(n) - k\, s(n-1) \tag{1}$$

donde $s'(n)$ denota la salida del filtro, $s(n)$ la muestra en el instante n y $s(n-1)$ la muestra anterior. En esta fórmula k es el coeficiente de preénfasis que debe estar dentro del rango $0 \leq k < 1$.

Generalmente el proceso de preénfasis se realiza para balancear la magnitud del espectro, es decir balancear los componentes considerados de frecuencias altas y bajas, atenuando las ultimas y resaltando las primeras (Loweimi, Ahadi, Drugman, y Loveymi, 2013). Con este preénfasis se elimina algún nivel de CD presente en las muestras y se acentúan las frecuencias altas, con lo cual se incrementa la precisión del reconocimiento. No obstante, Paliwal (1984) ha notado problemas en la detección de vocales ya que al atenuar las frecuencias bajas se presenta una leve degradación en su reconocimiento debido a que las vocales están acentuadas en la parte baja del espectro, sin embargo, menciona que en el caso del reconocimiento de voz para palabras aisladas con un número superior de consonantes el preénfasis incrementa la tasa de reconocimiento. Por lo tanto, recomienda aplicar el proceso de preénfasis dependiendo del tipo de vocabulario del sistema y de la presencia de vocales y consonantes.

2.1.1.2. Detección de actividad de voz

Verteletskaya y Sakhnov (2010) mencionan que el habla es una señal discontinua debido a que sólo envía información cuando se está hablando; a las secciones con habla presente se les llama secciones activas, mientras que a las pausas involuntarias entre pronunciaciones se les llaman secciones inactivas o silencios.

Para realizar el proceso de DAV o por sus siglas en inglés (VAD, *Voice Activity Detection*) se toma un vector de datos de la señal original, este vector se conoce como vector de características y es la entrada a una regla de decisión, la cual se encarga de clasificar este vector como sección activa o sección inactiva.

El proceso VAD es importante para el entrenamiento y detección en los sistemas modernos, sin embargo, aun presenta inconvenientes debido a problemas inherentes a la naturaleza de la señal, tal como frecuencias de las formantes y la variabilidad de la señal ya que la pronunciación no es uniforme. Según (Mustafa, Allen, y Evett, 2014) el proceso de VAD cumple un doble propósito: reducción en el número de elementos en la señal de audio y mejora de la clasificación de las señales a utilizar. Al reducir el número de elementos presentes en la señal a procesar se disminuye: el tamaño de los patrones

de audio, el tiempo de procesamiento de estos, el nivel de precisión del sistema en general e inclusive el espacio de almacenamiento.

Como parte de la fase de preprocesamiento, el proceso de Detección de Actividad por Voz es necesario para fases posteriores en donde se aplican algoritmos de mejora en el audio hablado. La Figura 2.2 muestra un ejemplo de aplicación de este tipo de algoritmo en donde se puede notar la reducción de elementos en el arreglo de aproximadamente 7500 a 4500 muestras. Lo cual muestra las ventajas de la aplicación de un algoritmo de este tipo.

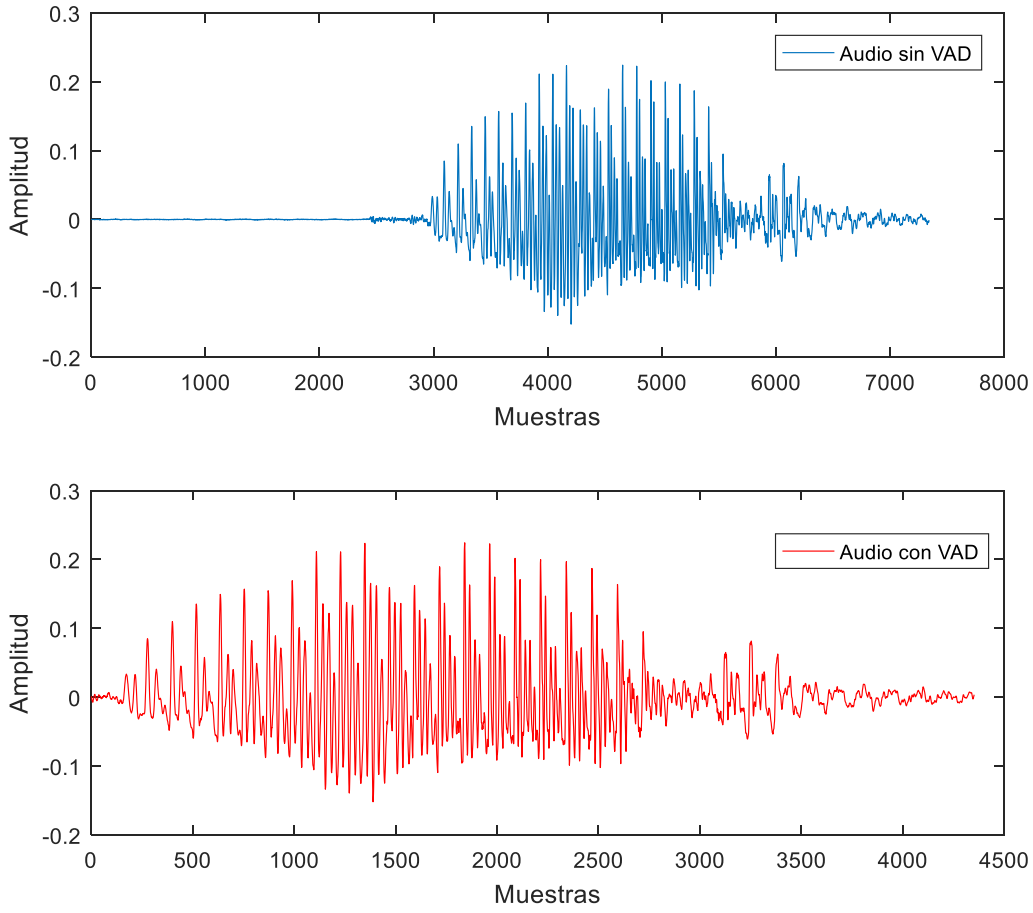


Figura 2.2. Ejemplo de aplicación de VAD.

Los algoritmos que se aplican en este proceso presentan dos variantes:

a) Detectores a base de energía: usan los siguientes métodos o combinaciones (Ramirez, Gorriz, y Segura, 2007):
   - Umbrales de energía.
   - Detección de tono.
   - Análisis espectral.

- Tasa de cruces por cero.
- Medida de periodicidad.
- Estadísticas de alto orden en el dominio residual de los LPC.

b) Detectores usando aprendizaje automático: utilizan modelos estadísticos y máquinas de soporte vectorial (Shin, Chang, y Kim, 2010) y esquemas basados en redes neuronales (Zhang y Wu, 2013), (Eyben, Weninger, Squartini, y Schuller, 2013), entre otros.

Los esquemas a) y b) ocupan la segmentación por ventana para aumentar la tasa de reconocimiento de fonemas y determinar de manera más robusta si es ruido, silencio o voz.

2.1.2. Segmentación y aplicación de máscara ventana

Una parte importante de este preprocesamiento consiste en el proceso de segmentación en ventanas (*Windowing)* es utilizado en el proceso de VAD y de extracción de características ya que es un componente vital para los sistemas de este tipo.

Para Loizou (2013) la transformada en tiempo discreto de Fourier (DTFT) necesita n valores de la señal $s(n)$. Debido a la naturaleza real de las señales que se presentan, estas señales deben ser reducidas a un número de elementos discretos. Matemáticamente esto equivale a truncar la función $s(n)$ a una duración finita, la cual se conoce como ventana (*window*), y afecta directamente al espectro de $s(n)$.

El proceso de segmentación en ventanas queda definido matemáticamente por la ecuación (2):

$$x_W(n) = s(n)w(n) \tag{2}$$

donde $x_W(n)$ es la señal resultante de aplicar la función ventana $w(n)$ a una señal $s(n)$.

Debido a que la voz no es una señal estacionaria y es variante en el tiempo, se tiene que procesar de una manera especial. En primer lugar, se asume que la voz consiste en un conjunto de fonemas y éstos tienen un periodo de duración en el cual permanecen invariantes (5-100 ms) (Mahkonen, 2017). Por lo tanto, si una señal no varía en sus frecuencias durante este intervalo de tiempo, se considera estacionaria en ese instante y, por lo tanto, se puede identificar el fonema hablado con mayor facilidad.

En la Figura 2.3 se muestra un ejemplo de segmentación y aplicación de máscara ventana.

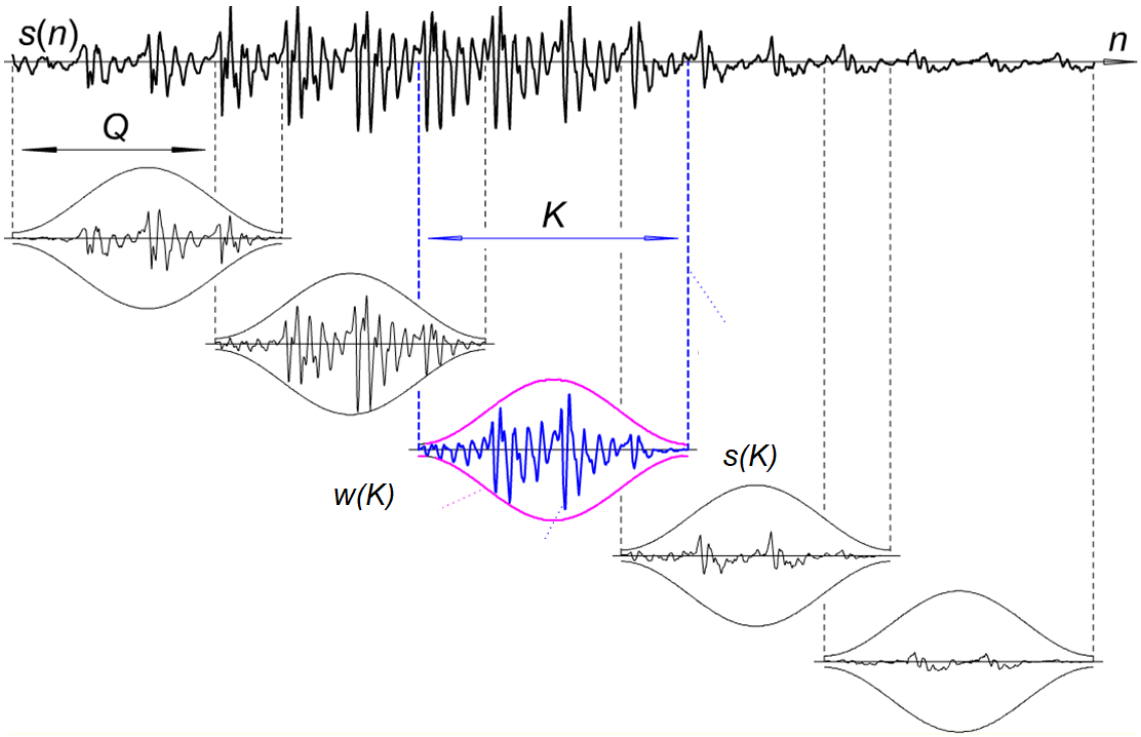


Figura 2.3. Visualización del proceso de Segmentación en marcos y aplicación de máscara ventana.

La señal $s(n)$, con el número total de muestras N, se divide en intervalos de longitud K que pueden estar o no imbricados. De esta manera, la señal original de N muestras se divide en marcos (*frames*) de K muestras, en los cuales se espera identificar el fonema hablado en ese intervalo de tiempo.

En la Figura 2.4 se muestra una representación aproximada de lo obtenido al segmentar el audio, en la misma se pueden observar cambios de los fonemas básicos, así mismo se muestra un ejemplo simple en el cual no se observa una coarticulación marcada. Para esto el audio se segmentó en marcos de 256 elementos usando una ventana de Hamming lo cual da como resultado marcos de 16 ms donde se observa lo mencionado en los párrafos previos.

A continuación, se genera la ventana por la cual pasarán las K muestras antes de iniciar cualquier proceso de extracción de características. Se recomienda usar ventanas con bordes suaves (Mahkonen, 2017) para evitar problemas en la fase de extracción de características.

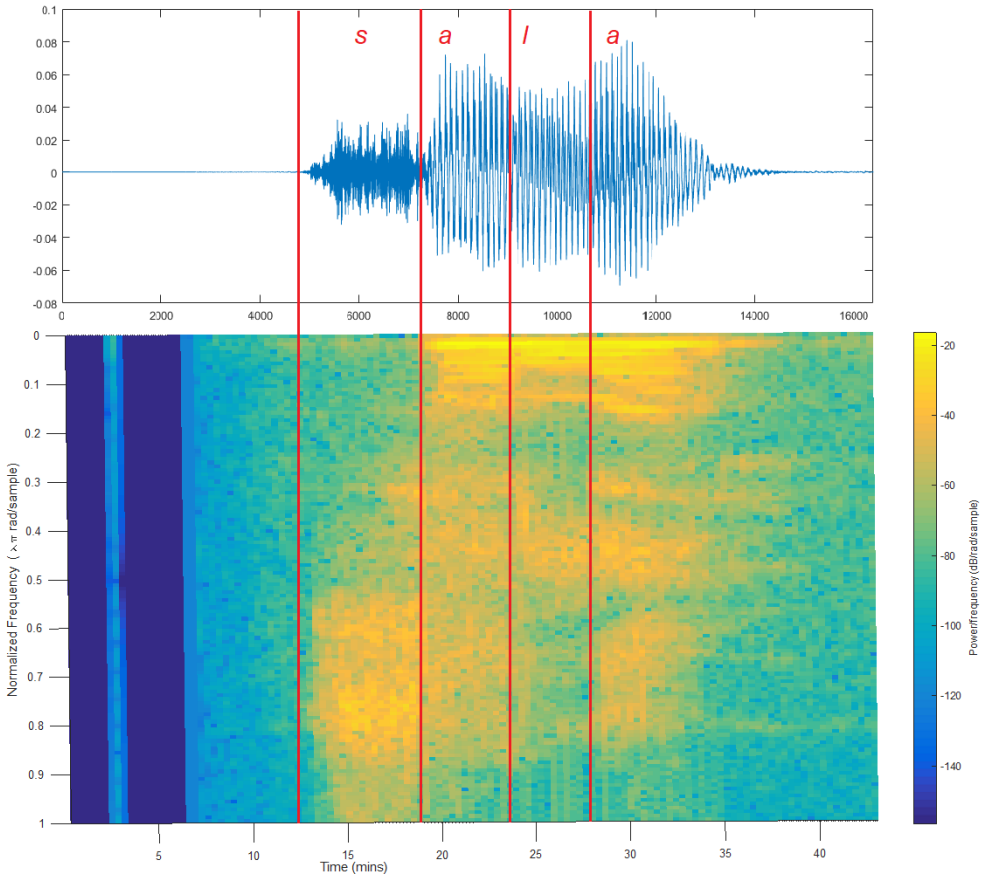


Figura 2.4. Espectrograma de la pronunciación "sala".

El proceso de segmentación en ventanas presenta el efecto de fuga espectral debido a que los lóbulos del espectro resultante de la ventana pueden aumentar la magnitud del espectro en ciertas regiones que no están relacionadas con el espectro original, esto se debe al tipo y forma de la ventana aplicada. No obstante, existen ventanas específicas que suavizan este efecto, tales como las ventanas Hamming y Hanning, las cuales reducen el ancho de banda del lóbulo central y la magnitud de los lóbulos laterales del espectro tal como se observa en la Figura 2.5, en la cual se puede observar que el resultado con ventana Hamming presenta menor ancho en los lóbulos.

Alsteris y Paliwal (2007) mencionan que la inteligibilidad resultante usando la ventana Hamming es superior en el procesamiento de las características de la voz. Cabe aclarar que esto es válido cuando sólo se usan características de magnitud del espectro y una duración de tiempo de 32 ms en la ventana. Lo anterior, comparado con una ventana rectangular que no tiene bordes suavizados, confirmando lo dicho por Mahkonen (2017).

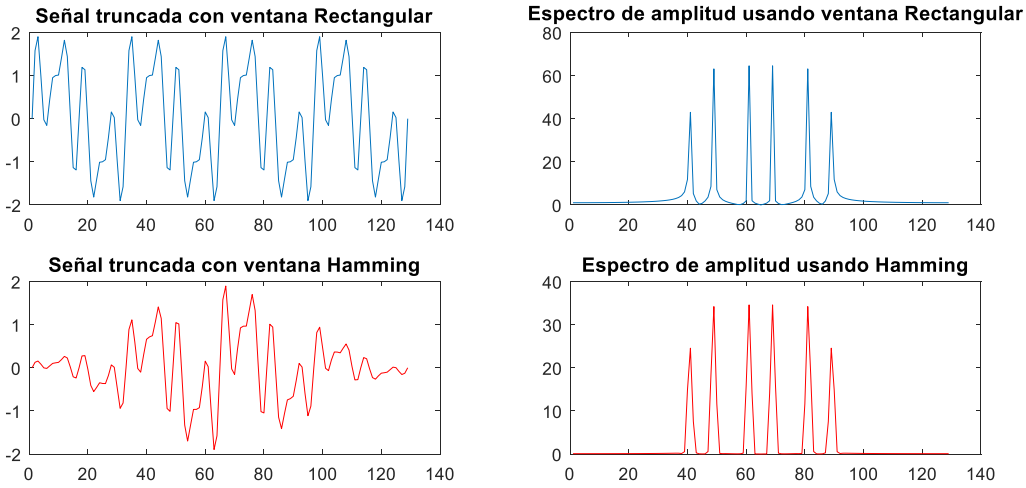


Figura 2.5. Comparativa de espectros obtenidos usando una ventana rectangular y una ventana Hamming. Se puede notar que el resultado con Hamming tiene menor ancho de lóbulos.

La función que genera la ventana de Hamming es presentada en la ecuación (3).

$$f(n) = .54 - .46\ \cos\left(\frac{2\pi(n-1)}{L-1}\right)\ para\ n = 1{,}2, \ldots . L \tag{3}$$

La Figura 2.6 muestra un ejemplo de este tipo de ventana, donde L es 41.

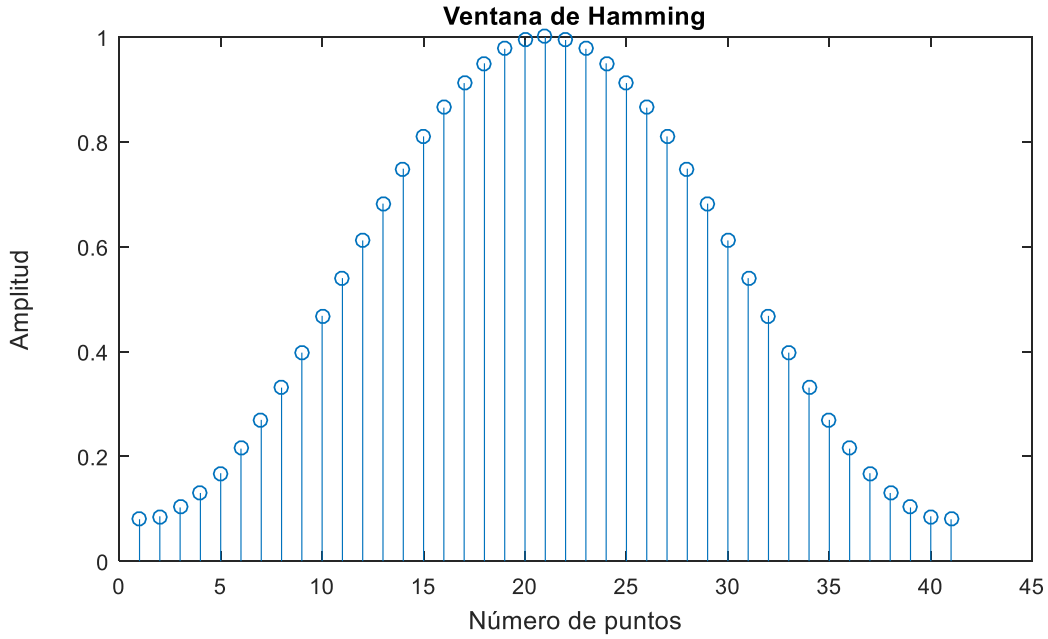


Figura 2.6. Ventana Hamming con L= 41.

Cada uno de los marcos de K muestras se multiplica por la ventana generada, creando los marcos sobre los cuales se trabajará en la fase de extracción de características.

2.1.3. Extracción de características

La extracción de características consiste en tomar un conjunto de datos de entrenamiento disponibles de entrada y transformarlos en otros datos que representen de mejor manera los datos iniciales. Estos datos procesados contienen ciertas propiedades discriminantes principales de los datos originales, un espacio de almacenamiento menor al original, mientras se preservan las características más importantes encontradas en el conjunto de datos originales.

En este procesamiento, en lugar de ocupar el conjunto de datos originales se ocupa una variante de menor dimensión y que posee las mismas características principales para su aplicación en alguna implementación de Aprendizaje Automático.

Dos de las técnicas más comunes para este tipo de extracción se realizan mediante el uso de Coeficientes de Predicción Lineal y los Coeficientes Cepstrales en las Frecuencias de Mel. En el primero se usa un enfoque basado en la producción de la voz y en el segundo un enfoque basado en un modelo perceptual.

2.1.3.1. Técnica de coeficientes de predicción lineal

La extracción mediante la técnica de coeficientes de predicción lineal (LPC, *Linear Prediction Coefficients*) se basa en la producción de la voz; para ello se modela la función de transferencia del tracto vocal mediante un filtro que contiene puros polos y cuya función de transferencia tiene la forma mostrada en (4).

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-i}} \tag{4}$$

donde p denota el número de polos del filtro y a_0=1; se seleccionan los coeficientes $\{a_i\}$ restantes de manera que éstos minimicen la suma del error cuadrático medio durante el marco que se esté procesando en cierto instante.

Se implementa un método de autocorrelación que lleve a cabo este proceso de optimización y permita obtener los coeficientes más útiles para el procesamiento. A continuación, se procesa cada uno de los marcos obtenidos en la etapa anterior para obtener los coeficientes que permitan el procesamiento de la voz de manera precisa.

2.1.3.2. Coeficientes Cepstrales de las Frecuencias de Mel

Otra técnica utilizada para extraer las características de los fonemas hablados consiste en los Coeficientes Cepstrales de las Frecuencias de Mel (MFCC, *Mel Frequency Cepstral Coefficients*). En esta técnica se procesan los fonemas para obtener resultados para cada uno de los marcos (Jang, 2017), se considera un enfoque perceptual basado en la noción de que el oído humano es más sensible a las bajas frecuencias, desvaneciéndose esta sensibilidad conforme aumenta la frecuencia que se escucha, siendo ésta más sensible a frecuencias menores a 5 kHz (Gelfand, 2016).

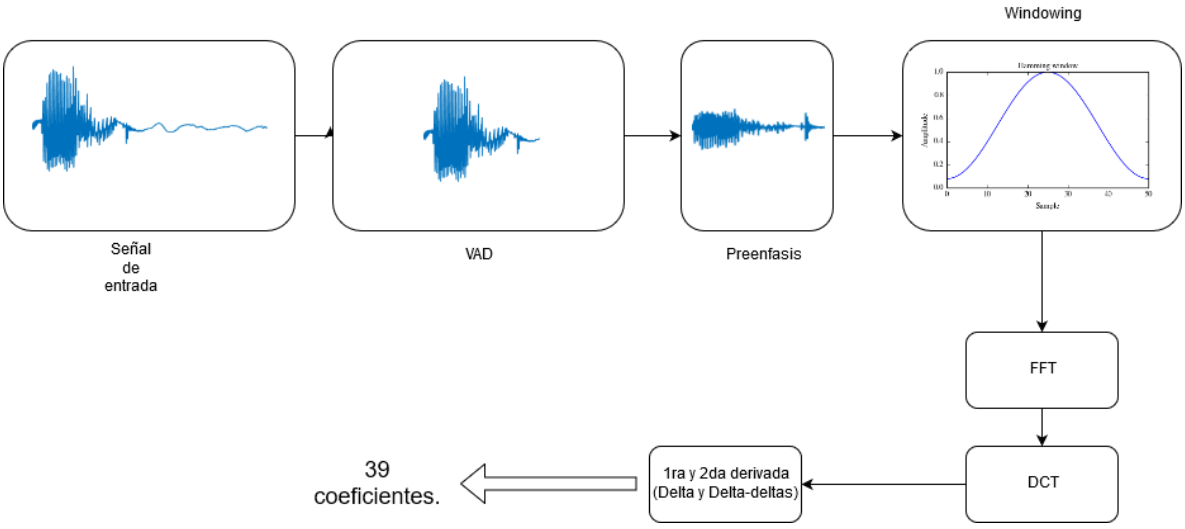


Figura 2.7 Esquema de extracción de características usando MFCC.

MFCC es una de las técnicas más usadas y conocidas para la extracción de características en el reconocimiento de voz. Ésta ha presentado una ventaja en ciertas condiciones y aplicaciones en diferentes ámbitos tal como: reconocimiento de usuarios, búsqueda de información de música y reconocimiento de voz. Por lo anterior, en 2003 el Instituto Europeo de Normas de Telecomunicaciones (ETSI) lo declaró como estándar de extracción de características para sistemas de reconocimiento de voz distribuidos (European Telecommunications Standards Institute, 2018). Así mismo, esta técnica produce resultados satisfactorios tal como lo muestran los estudios de (Kishore & P., 2016) y (Taabish, Anand, & Sandeep, 2014).

A continuación, se describe de manera general, el procedimiento MFCC. Sea f el marco que se desea analizar, a este marco se aplica la transformada de Fourier, donde $F\{f(t)\}$ es la transformada de Fourier de la función f(t) y $F^{-1}\{f(j\omega)\}$ es la inversa del resultado obtenido en la transformada de Fourier de la función $f(t)$. Como resultado se

obtiene su transformada de Fourier de $F\{f(t)\}$. El cuál es el valor del espectro de la señal a procesar.

Una vez obtenida la transformada del marco, se procede a diseñar un banco de filtros en la escala de Mel. Donde la escala de Mel es una escala musical perceptual de tonos juzgados como intervalos equiespaciados por parte de observadores, la cual está dada por (5).

$$Mel(Frec) = 1127 \ln\left(1 + \frac{Frec}{700}\right) \tag{5}$$

A continuación, se calculan ventanas triangulares que funcionarán como filtros, éstas pueden quedar sobrepuestas o no. La frecuencia central de cada ventana se obtiene usando el proceso inverso dado por (6).

$$Frec(Mel) = 700\left(e^{\frac{Mel}{1127}} - 1\right) \tag{6}$$

Cabe mencionar que Mel denota una frecuencia en Mels y $Frec$ una frecuencia en Hertz. Todas estas conversiones se realizan para obtener la escala deseada del filtro.

Cada uno de los valores que se calculan para el diseño de estos filtros tiene su correspondiente valor en frecuencia, lo cual da como resultado un banco de filtros superpuesto con una escala logarítmica (Véase Figura 2.8).

Se procede a filtrar el resultado de $F\{f(t)\}$ para obtener coeficientes con las señales representativas de las frecuencias que se desean. Una vez obtenidos dichos coeficientes en escala de Mel, se genera el logaritmo de cada uno de estos coeficientes.

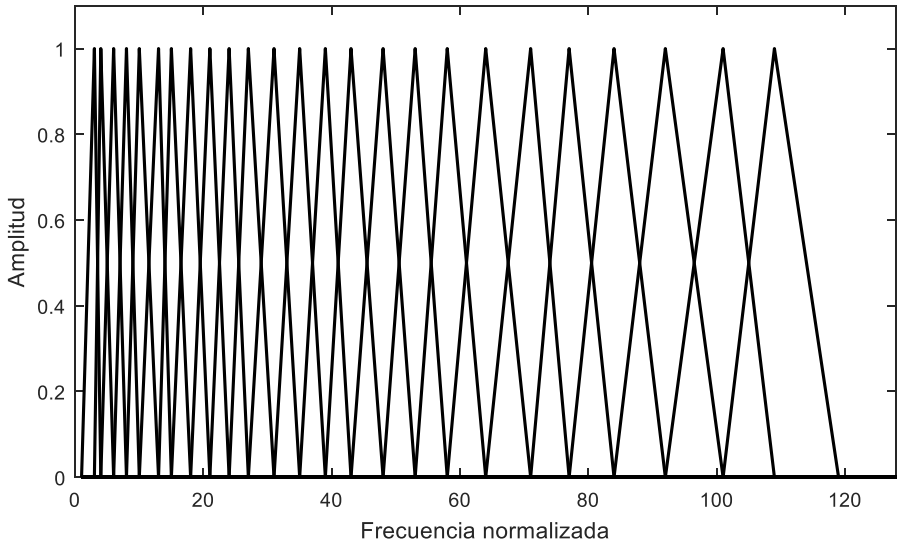


Figura 2.8. Banco de filtros usando ventanas triangulares.

Sean C_{f_n} los coeficientes obtenidos para cada uno de los filtros triangulares n, aplicando lo anterior se obtiene lo mostrado en (7).

$$log_{10}(C_{f_n}) = log\ C_{f_n} \tag{7}$$

Finalmente, se aplica la Transformada Discreta del Coseno (DCT) denotada por $DCT\{F(x)\}$ para obtener coeficientes no correlacionados que serán utilizados para cálculos posteriores, este paso simula la aplicación de PCA. Para lograr esto, primero se elimina el primer coeficiente de la transformación anterior y, en la mayoría de los casos, se eliminan los coeficientes mayores al orden 13.

La eliminación del primer coeficiente se realiza para intercambiarlo con la suma del valor absoluto de la energía E del banco de filtros, tal como muestra la ecuación (8).

$$E = \sum_{n=1}^{N} C_{fn} \tag{8}$$

donde C_{fn} indica el valor de energía en el n filtro y representa el primer valor en los MFCC.

El segundo paso, consiste en eliminar los coeficientes mayores al orden 13, esto se realiza debido a las propiedades de compresión inherentes a la DCT, lo cual permite reducir el número de características sin impactar el rendimiento de los coeficientes obtenidos.

Con los coeficientes resultantes se prosigue al cálculo de las Deltas y las Deltas-Deltas (aceleración), las cuales consisten en una serie de diferencias que sirven para captar la condición no estacionaria de las señales en el habla, es decir, para capturar sus componentes dinámicos y con esto se mejora el rendimiento del sistema en general. Existen varias formas de calcular estos Deltas, sin embargo, la más usada y sencilla es la que se muestra en la formula (9) (Young, et al., 2017).

$$d_t = \frac{C_{n+\theta} - C_{n-\theta}}{2\theta} \tag{9}$$

donde θ representa el valor de ventana para el Delta número t.

Generalmente, las características resultantes quedan de la siguiente manera:

- 1 valor de energía.
- 12 coeficientes cepstrales.
- 1 delta de energía.

- 12 deltas de los coeficientes cepstrales.
- 1 aceleración de la energía.
- 12 aceleraciones de los coeficientes cepstrales.

Obteniéndose de esta manera el vector de características con 39 elementos que será usado para el reconocimiento de voz.

No existe sólo un esquema para la obtención de estos coeficientes y, de manera general, existen muchos parámetros que pueden modificar el rendimiento de la extracción de características. Esto ha derivado en múltiples implementaciones del método, las cuales difieren principalmente en: número de filtros, ancho de banda del espectro de audio, forma de los filtros, espacio entre los filtros y manera en la cual éstos distorsionan el espectro (Zheng, Zhang, y Song, 2001). Estos parámetros pueden ser modificados para obtener la configuración óptima (Lawson, et al., 2011). Existen otros parámetros importantes tales como: rango de frecuencia, selección del número de Coeficientes Cepstrales, número de deltas y de deltas-deltas, los cuales juegan un papel importante en la fase de clasificación.

De acuerdo con un estudio realizado por Ganchev, Fakotakis y Kokkinakis (2005) existen tres técnicas base para obtener los MFCC, los cuales se describen a continuación.

2.1.3.2.1. MFCC Mermelstein

Esta técnica fue propuesta por Davis y Mermelstein (1980), asume una frecuencia de muestreo de 10 kHz y un ancho de banda del habla de 0 a 4.6 kHz. Así mismo, hace uso de 20 filtros triangulares, con la misma amplitud máxima, en los cuales se procesan las energías del espectro de voz obtenidas mediante una transformada de tiempo corto de Fourier de n puntos. Estos filtros están distribuidos de manera perceptual, es decir, enfocados en las frecuencias bajas, mientras generan las frecuencias centrales con la siguiente ecuación dependiendo la frecuencia central f_c del filtro i siguiendo la ecuación (10).

$$f_{ci} = \begin{cases} 100\ i\ para\ i = 1,2,3 \dots 10 \\ f_{c10}\ 2^{0.2(i-10)}\ para\ i = 11,12, \dots 20 \end{cases} \quad (10)$$

Considerando que los filtros están distribuidos entre 0 y 4.6 kHz, se calculan las máscaras de cada filtro triangular con ayuda de (11).

$$H_i(k) = \begin{cases} 0\ para\ k < f_{bi-1} \\ \dfrac{(k - f_{bi-1})}{(f_{bi} - f_{bi-1})}\ para\ f_{bi-1} \leq k \leq f_{bi} \\ \dfrac{(f_{bi+1} - k)}{(f_{bi+1} - f_{bi})}\ para\ f_{bi} \leq k \leq f_{bi+1} \\ 0\ para\ k > f_{bi+1} \end{cases} \quad (11)$$

donde cada filtro está denotado por H_i , i denota el i-esimo filtro triangular y k es el valor para cada uno de los correspondientes puntos de la transformada de Fourier a la cual se va a filtrar. Entonces, f_{bi} que indica cuando el filtro se hace cruce por cero está dado por la frecuencia de muestreo F_s y por el número de puntos en la transformada de Fourier denotado por N. Con esto el valor f_{bi} queda expresado por la ecuación (12).

$$f_{bi} = \left(\frac{N}{F_s}\right) Frec\left(Mel(f_{inicial}) + i\frac{Mel(f_{final}) - Mel(f_{inicial})}{M+1}\right) \quad (12)$$

donde f_{final} es la frecuencia más alta en el ancho de banda y $f_{inicial}$ es la frecuencia más baja en el ancho de banda, a su vez M denota el número de filtros triangulares, Mel indica la conversión entre unidades Hertz a Mels, y Frec la conversión de Mels a Hertz.

Esta técnica utiliza el valor absoluto del espectro, el cual no cumple con la relación de Parseval que indica que la suma de los cuadrados es la energía y no la suma del valor absoluto. Cabe señalar que, aunque esta definición no se cumpla en varios algoritmos, sigue siendo utilizada de igual manera.

El espectro de la ventana pasa por cada uno de los bancos triangulares obteniéndose la energía en cada uno de ellos, finalmente, se aplica la DCT y se sigue con el cálculo de los marcos restantes.

2.1.3.2.2. MFCC HTK

La técnica MFCC HTK fue descrita por Young (1995) en el Cambridge HMM Toolkit (HTK) y es una de las variantes más usadas para reconocimiento de voz. Esta técnica se compone de 24 filtros distribuidos en un ancho de banda del espectro de voz, situado entre 0 y 8 kHz, en este caso el muestreo de la señal de audio se realiza a una frecuencia mayor de 16 kHz. En esta implementación el número de filtros y el ancho de banda del espectro de voz puede ajustarse permitiendo variaciones en la configuración.

MFCC HTK hace uso de la definición de la frecuencia de Mel para obtener un intervalo de frecuencias llamado Δf con unidades Mel, el cual se expresa la separación

de cada uno de los filtros triangulares, a su vez cada uno de estos filtros tiene la misma magnitud tal como se presenta en el caso de Mermelstein. La ecuación (13) muestra cómo obtener Δf.

$$\Delta f = \frac{Mel(f_{final}) - Mel(f_{inicial})}{M + 1} \tag{13}$$

Una vez obtenido este Δf, se procede a calcular la frecuencia central para el i-ésimo filtro en unidades Mel de cada uno de los M filtros usando la ecuación (14).

$$m_{ci} = Mel(f_{inicial}) + i\,\Delta f \tag{14}$$

A continuación, se prosigue a la conversión de las frecuencias de Mel a Hertz para implementar el banco de filtros, mediante la ecuación (15):

$$f_{ci} = Frec(m_{ci}) \tag{15}$$

Una vez obtenidos todos los parámetros se procede al cálculo de los filtros triangulares, los cuales se obtienen mediante la ecuación (11) usada previamente por el esquema de Mermelstein.

Finalmente, la energía se comprime usando el logaritmo del valor absoluto. El espectro de la ventana pasa por cada uno de los bancos triangulares obteniendo la energía en cada uno de ellos; por último, se aplica la DCT y se sigue con el cálculo de los marcos restantes.

2.1.3.2.3. MFCC Slaney

Esta técnica se introdujo por primera vez en el Auditory Toolbox para MATLAB y se describió en 1998 por Slaney (2018). Se asume una frecuencia de muestreo de 16 kHz y un ancho de banda del espectro de voz de 133 a 6854 Hz. Se utilizan 40 filtros triangulares de igual área, los primeros 13 filtros están distribuidos de manera uniforme en el rango de 200 a 1000 Hz con una diferencia de 66.67 Hz, mientras que los siguientes 27 filtros están distribuidos logarítmicamente en el rango de 1071 a 6400 Hz usando la ecuación (16) para obtener la diferencia.

$$logStep = exp\left(\frac{ln\left(\frac{f_{c40}}{1000}\right)}{nlF}\right) \tag{16}$$

donde f_{c40} es la frecuencia central de filtro 40 y nlF es el número de filtros logarítmicos.

Los filtros triangulares están definidos como lo muestra (17).

$$H_i(k) = \begin{cases} 0\ para\ k < f_{bi-1} \\ \dfrac{2(k - f_{bi-1})}{(f_{bi} - f_{bi-1})(f_{bi+1} - f_{bi-1})}\ para\ f_{bi-1} \leq k \leq f_{bi} \\ \dfrac{2(f_{bi+1} - k)}{(f_{bi+1} - f_{bi})(f_{bi+1} - f_{bi-1})}\ para\ f_{bi} \leq k \leq f_{bi+1} \\ 0\ para\ k > f_{bi+1} \end{cases} \quad (17)$$

Debido a la presencia de la normalización del área, dada por el coeficiente mostrado en (18).

$$\frac{2}{(f_{bi+1} - f_{bi-1})} \quad (18)$$

La suma del banco de filtros de los coeficientes es unitaria y por lo tanto cada uno de los filtros satisface la condición mostrada en la ecuación (19).

$$\sum_{k=1}^{N} H_i(k) = 1 \quad (19)$$

Finalmente, la energía se comprime usando el logaritmo del valor absoluto. El espectro de la ventana pasa por cada uno de los bancos triangulares obteniendo la energía en cada uno de ellos; por último, se aplica la DCT y se sigue con el cálculo de los marcos restantes.

2.1.4. Algoritmos de comparación de patrones DTW

El algoritmo de Distorsión Dinámica del Tiempo (*Dynamic Time Warping* o DTW por sus siglas en inglés) es un algoritmo empleado para encontrar el alineamiento óptimo entre dos señales; se utiliza para comparar dos secuencias de valores que están desalineadas o que varían en velocidad sobre el eje del tiempo bajo ciertas restricciones (Dhingra, Nijhawan y Pandit, 2013). Bajo este esquema, se realiza el alineamiento de forma que las señales son distorsionadas o deformadas de manera no lineal para hacerlas coincidir (Sakoe y Chiba, 1978).

En la Figura 2.9 se pueden observar señales que son similares pero que contienen diferencias en el eje x.

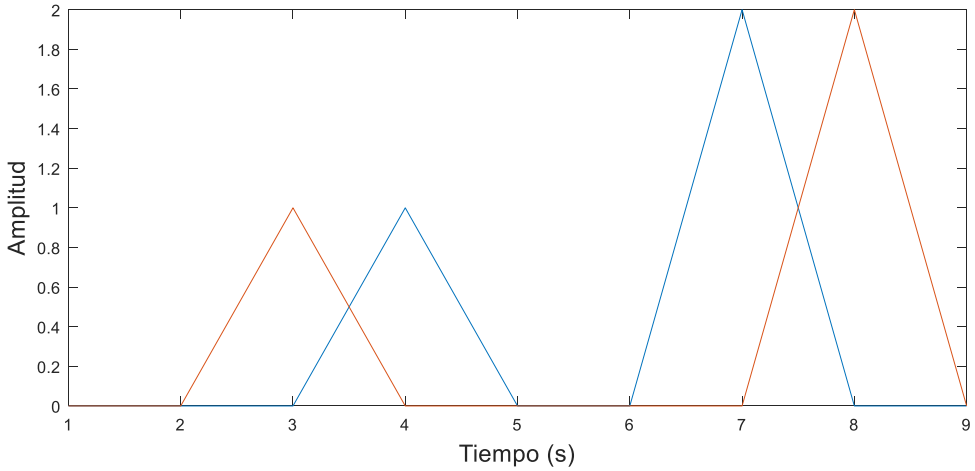


Figura 2.9. Señales similares con diferencias en la velocidad en el eje x.

Originalmente, este algoritmo fue diseñado para comparar patrones en el reconocimiento de voz, no obstante, ha encontrado cierto impacto en la minería de datos (*data mining*) y en la búsqueda y recuperación de información, así mismo, ha sido aplicado con éxito a la adaptabilidad de las deformaciones de los datos dependientes del tiempo (Müller, 2007).

Su funcionamiento es el siguiente: mediante el uso de la programación dinámica se realiza un efecto de normalización de tiempo, para esto se modela la fluctuación en el eje del tiempo usando algunas propiedades específicas, como muestra la Figura 2.10.

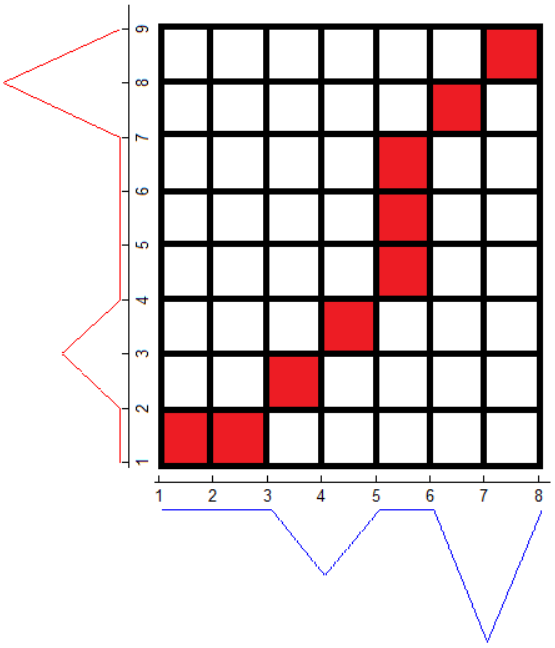


Figura 2.10. Matriz mostrando el Warping Path entre dos señales.

Se eliminan las fluctuaciones en el eje del tiempo mediante la deformación del eje del tiempo de una señal con respecto de la otra. Después de la alineación, se calcula la distancia de tiempo normalizada como la mínima distancia residual entre ambas. Este último proceso de minimización es llevado a cabo mediante el uso de la programación dinámica y el cálculo del *Warping Path*.

Este algoritmo presenta dos restricciones que deben cumplirse para un funcionamiento óptimo: a) ambas señales deben estar muestreadas de manera constante y b) ambas deben ser muestreadas con la misma frecuencia de muestreo. Supóngase que existen dos señales en el tiempo, A y B:

$$A = a_1, a_2, a_3, a_4, a_i \dots a_I$$

$$B = b_1, b_2, b_3, b_4, b_j \dots b_J$$

donde I y J denotan el tamaño de las señales A y B, pudiendo I y J ser diferentes. La Figura 2.11 muestra que ambas señales ocupan un eje de un plano, el cual está conformado por cada una de las combinaciones entre las señales a_i y b_j, estas combinaciones conforman la matriz de costo $C(i,j)$, la cual se calcula con el resultado obtenido por la medida de disimilitud o la distancia entre a_i y b_j,denotada por $d(i,j)$.

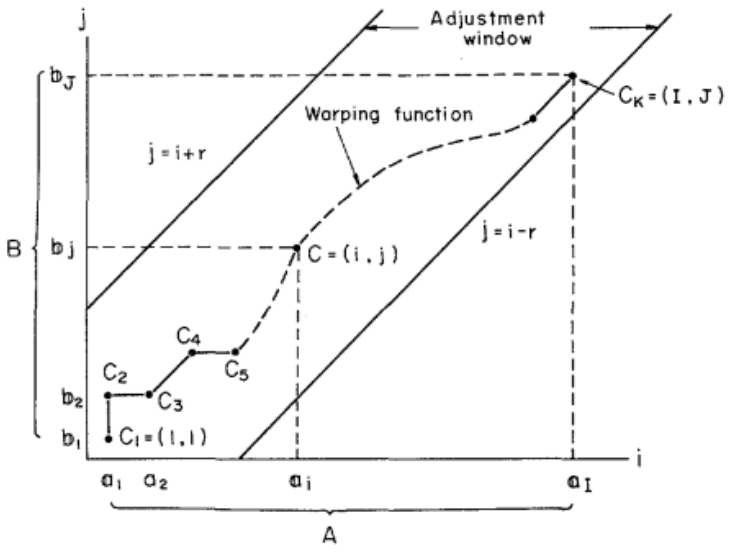


Figura 2.11. Visualización del cálculo del costo del algoritmo DTW.

Se emplea una función de distorsión para realizar el cálculo de la menor distancia entre las dos señales a comparar, la cual está definida por (20).

$$F = c(1), c(2), c(k) \dots\ c(K) \tag{20}$$

donde $c(k) = (i(k), j(k))$ es la posicion de la matriz de costo respecto al k-ésimo paso en la búsqueda del término con la menor distancia adyacente.

La suma de pasos $c(k)$ de la función de distorsión se transforma en la distancia entre las dos señales, y la ecuación (21) para obtener la distancia.

$$E(F) = \sum_{k=1}^{K} d\big(c(k)\big)\, w(k) \tag{21}$$

Nótese que en la sumatoria se incluye una ponderación para esta suma, esta ponderación se agrega para brindar flexibilidad al cálculo, así mismo, es necesario que los pesos $w(k)$ sean positivos o 0. A su vez, el *Warping Path* debe cumplir con tres condiciones necesarias:

- Condición de frontera: $i(1) = 1\,, j(1) = 1\,, i(K) = I\ y\ j(K) = J$.
- Condición de monotonía: $i(k\ -\ 1) \leq\ i(k)\ y\ j(k\ -\ 1)\ \leq\ j\ (\ k\ )$.
- Condición de continuidad: $i(k) -\ i(k\ -\ 1) \leq 1\ y\ j(k)\ -\ j(k\ -\ 1) \leq\ 1$.

La Figura 2.12 muestra ejemplos de las condiciones necesarias, a) Cumple las tres condiciones, b) Incumple la condición de frontera, c) Incumple condición de monotonía y d) Incumple condición de continuidad.

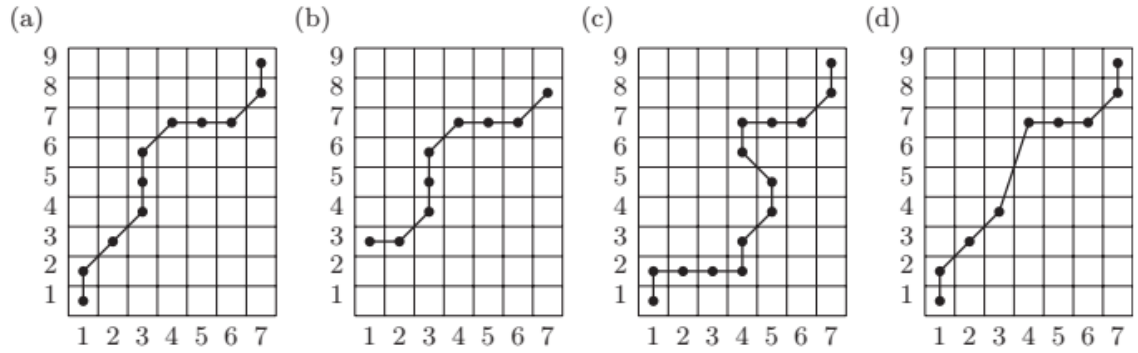


Figura 2.12. Condiciones necesarias de Warping Path.

Con base en las condiciones dadas, como resultado se obtiene la condición de escalón básica del *Warping Path*, la cual consta de las siguientes ecuaciones, en específico esta versión obtenida es el escalón usado por White y Neely (1976) se muestra en la ecuacion (22).

$$c(k-1) = \begin{cases} \big(i(k), j(k-1)\big) \\ \big(i(k), j(k)\big) \\ \big(i(k-1), j(k)\big) \end{cases} \tag{22}$$

Enseguida se procede al cálculo de la distancia haciendo uso de las distancias locales contenidas en la matriz C, para esto se calcula la matriz de costo acumulado que regresa el resultado de la distancia y en algunas variantes el *Warping Path* que contiene la ruta óptima de alineación entre las dos señales a comparar.

La matriz de costo acumulada se obtiene iterando la ecuación de recurrencia (23) para todos los i y para todos los j:

$$C_{acum}(1,1) = 0$$

$$C_{acum}(i,j) = d(i,j) + min\begin{bmatrix} C_{acum}(i,j-1) \\ C_{acum}(i-1,j) \\ C_{acum}(i-1,j-1) \end{bmatrix} \tag{23}$$

donde $d(i,j)$ es la medida de disimilitud entre el elemento a_i y el elemento b_j, obtenida de la matriz de costo local. Finalmente, la distancia final se almacena en la posición $C_{acum}(I,J)$.

El cálculo del DTW tiene varias configuraciones, las cuales incluyen opciones de restricciones de escalón y restricciones globales. Otro factor que puede ayudar al desempeño es la selección de métricas y medidas de disimilitud. A continuación, se presentan las restricciones más comunes, las cuales ayudan a controlar el *Warping Path* para mejorar el cómputo de este algoritmo y los resultados obtenidos.

2.1.4.1. Restricciones de escalón

Las restricciones de escalón (*Slope constraints*) deben cumplir con las tres condiciones de frontera, monotonía y continuidad, para que el *Warping Path* sea lo más optimo posible.

En la Figura 2.13 se muestran ejemplos de *Warping Path:* a) y b) presentan repetición de pasos en x y en y formando de esta manera un *Warping Path* que no ajusta debidamente la alineación de ambas señales. En el inciso c) se presenta un *Warping Path* que presenta una ruta no tan brusca y una mejor alineación entre señales.

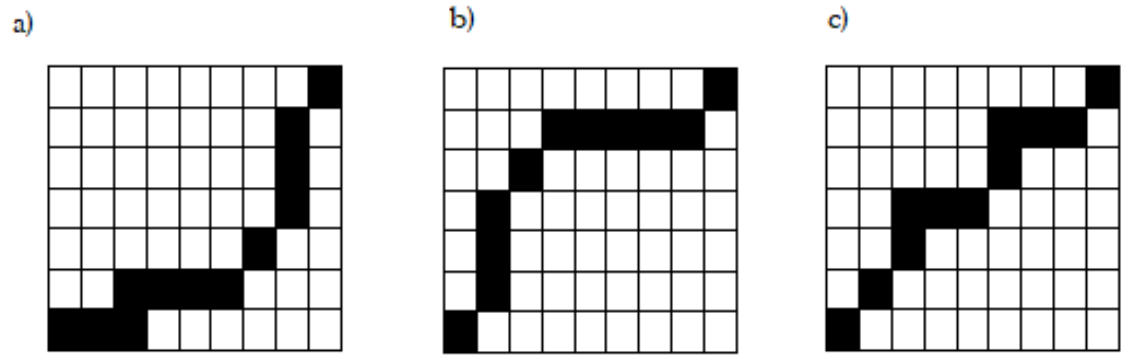


Figura 2.13. Diferentes tipos de Warping Path obtenidos debido a restricciones locales.

Por lo tanto, la selección de las restricciones de escalón debe ayudar a mejorar el desempeño del algoritmo DTW y obtener mejores resultados en la distancia final. Para esto, las restricciones más conocidas se describen a continuación.

La restricción Sakoe-Chiba con su respectivo valor de escalón y su condición de simetría. El valor de escalón hace referencia al número de pasos repetidos en x y en y que serán permitidos, para esto el valor de escalón está dado por la relación del número máximo de m movimientos verticales u horizontales consecutivos a partir de los cuales el algoritmo procederá a avanzar diagonalmente n veces. La ecuación (24) permite obtener el valor de escalón P.

$$P = \frac{n}{m} \tag{24}$$

Si el escalón seleccionado es simétrico esto implica que (25) se cumple:

$$C_{acum}(I,J) = C_{acum}(J,I) \tag{25}$$

en caso contrario el escalón es asimétrico. En la Tabla 2.1, se muestran las combinaciones más comunes propuestas por (Sakoe & Chiba, 1978).

La restricción White & Neely (1976) es la restricción de escalón más sencilla y su ecuación de recurrencia se muestra en la Tabla 2.2.

La restricción Itakura (1975) restringe el número de movimientos horizontales a 1, la Tabla 2.3 presenta su ecuación de recurrencia.

Tabla 2.1. Restricciones de escalón de Sakoe-Chiba.

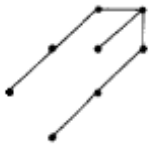
P	Guía visual	Tipo	Ecuación de recurrencia $C_{acum}(i,j)$
0		Simétrica	$min\begin{bmatrix} C_{acum}(i,j-1)+d(i,j) \\ C_{acum}(i-1,j)+d(i,j) \\ C_{acum}(i-1,j-1)+2d(i,j) \end{bmatrix}$
		Asimétrica	$min\begin{bmatrix} C_{acum}(i,j-1) \\ C_{acum}(i-1,j)+d(i,j) \\ C_{acum}(i-1,j-1)+d(i,j) \end{bmatrix}$
1		Simétrica	$min\begin{bmatrix} C_{acum}(i-1,j-2)+2d(i,j-1)+d(i,j) \\ C_{acum}(i-1,j-1)+2d(i,j) \\ C_{acum}(i-2,j-1)+2d(i-1,j)+d(i,j) \end{bmatrix}$
		Asimétrica	$min\begin{bmatrix} C_{acum}(i-1,j-2)+(d(i,j-1)+d(i,j))/2 \\ C_{acum}(i-1,j-1)+d(i,j) \\ C_{acum}(i-2,j-1)+d(i-1,j)+d(i,j) \end{bmatrix}$
2		Simétrica	$min\begin{bmatrix} C_{acum}(i-2,j-3)+2d(i-1,j-2)+2d(i,j-1)+d(i,j) \\ C_{acum}(i-1,j-1)+2d(i,j) \\ C_{acum}(i-3,j-2)+2d(i-2,j-1)+2d(i-1,j)+d(i,j) \end{bmatrix}$
		Asimétrica	$min\begin{bmatrix} C_{acum}(i-2,j-3)+2(d(i-1,j-2)+d(i,j-1)+d(i,j))/3 \\ C_{acum}(i-1,j-1)+d(i,j) \\ C_{acum}(i-3,j-2)+d(i-2,j-1)+d(i-1,j)+d(i,j) \end{bmatrix}$

Tabla 2.2. Restricción de White y Neely.

Guía visual	Ecuación de recurrencia $C_{acum}(i,j)$
	$d(i,j)+min\begin{bmatrix} C_{acum}(i,j-1) \\ C_{acum}(i-1,j) \\ C_{acum}(i-1,j-1) \end{bmatrix}$

Tabla 2.3. Restricción de Itakura.

Guía visual	Ecuación de recurrencia $C_{acum}(i,j)$
	$d(i,j)+min\begin{bmatrix} C_{acum}(i-1,j)+\alpha\, d(i,j) \\ C_{acum}(i-1,j-1)+d(i,j) \\ C_{acum}(i-1,j-2)+d(i,j) \end{bmatrix}$ Donde $\alpha=\infty$ si $i(k-1)=i(k-2))$ $\alpha=1$ si $i(k-1)\neq i(k-2))$

2.1.4.2. Restricciones globales

La selección de restricciones globales (g*lobal constraints*), también llamadas funciones ventana, ayuda a restringir el cálculo de la matriz C_{acum} a ciertas regiones, con lo cual se acelera el cálculo de las distancias, se mejora el tiempo de ejecución del

algoritmo, y se evita tomar alineaciones mal realizadas (*Pathologic Warping Paths*) debido a la selección de la restricción de escalón.

A continuación, se presentan las restricciones globales más usadas (véase Figura 2.14):

a) Banda Sakoe & Chiba (1978): presenta un efecto de distorsión cuando las dos señales difieren en tamaño, quedando deforme la distribución de la zona activa respecto a la señal a comparar.
b) Paralelogramo Itakura (1975): presenta dos valores de escalón, el más agudo de $\frac{1}{2}$ y el otro de 2, formando por esta razón un paralelogramo.
c) Ventana de Paliwal (Paliwal, Agarwal, y Sinha, 1982): es una modificación al esquema original de Sakoe-Chiba, siendo éste simétrico sin deformarse como el original.

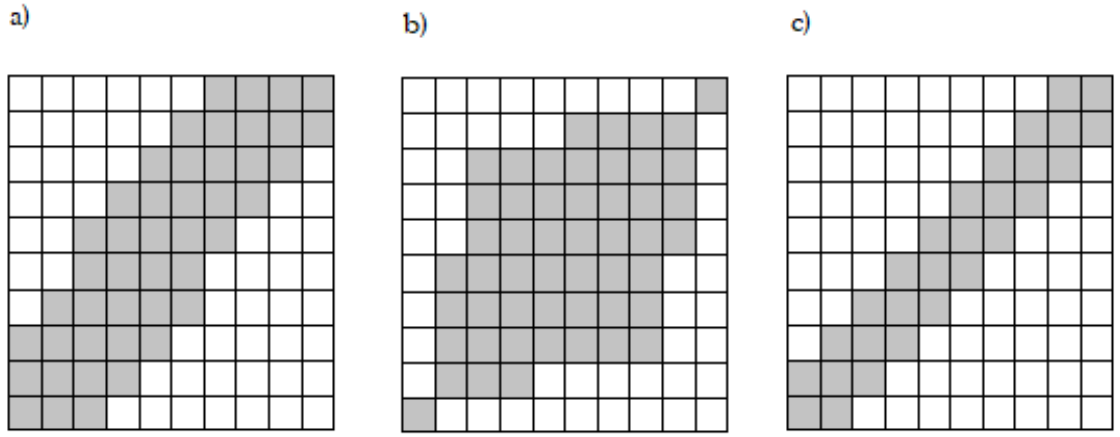


Figura 2.14. Tipos de restricciones globales para el algoritmo DTW.

2.1.4.3. Ponderaciones locales en la restricción de escalón

Como se mencionó en la explicación del algoritmo, se puede hacer uso de ponderaciones $w(k)$, las cuales pueden ayudar a favorecer ciertos movimientos ya sean horizontales, verticales o diagonales, con el fin de obtener mejores resultados. Se debe realizar este paso de manera heurística hasta obtener los mejores resultados.

2.1.4.4. Medidas de disimilitud

Para el cálculo de las matrices de costo, en el algoritmo se hace uso de medidas de disimilitud, es decir, el algoritmo favorece éstas debido a que entre más similares sean los objetos a_i y b_j menor será el valor obtenido al aplicar $d(i,j)$; para esto, existen dos variantes comúnmente usadas, las cuales consisten en la distancia Euclidiana y la distancia Manhattan.

Es preferible el uso de la distancia Euclidiana debido a su precisión, no obstante, se suele usar la distancia Manhattan debido a que ésta consume menor tiempo de cómputo respecto a la distancia Euclidiana, ya que la distancia Euclidiana hace uso de operaciones de potenciación y de raíz cuadrada para obtener el resultado. Cabe remarcar que en la literatura no se encuentran más medidas de disimilitud que sean usadas para este tipo de problemas, lo cual deja espacio abierto para el estudio del efecto de dichas medidas en este algoritmo.

2.2. Interfaces de usuario por voz

Una Interfaz de Usuario por Voz (VUI, por sus siglas en inglés) es una aplicación en la cual una persona interactúa utilizando el lenguaje hablado para realizar acciones dentro de dicha aplicación. Sus elementos más comunes son: *prompts*, gramáticas y lógica de diálogo, también llamada flujo de llamada.

Los *prompts* son mensajes de sistema que son grabados o sintetizados cuando se realiza una llamada al sistema y que son reproducidos cuando el usuario realiza una acción. Para esto se debe conocer cada una de las posibles acciones del usuario y la respuesta que tendrá el sistema mediante los *prompts*. El sistema sólo reconoce las opciones de diálogo válidas, las cuales se declaran en las gramáticas a utilizar. Para esto, se necesita la aplicación de una lógica de diálogo la cual decide cómo se va a reaccionar con base en las acciones del usuario, ya que, dependiendo del conjunto de respuestas, el sistema realizará alguna acción predeterminada.

2.2.1. Metodología de diseño de las VUI

La metodología y principios de diseño de las VUI se superponen substancialmente con otras metodologías de software, no obstante, imponen otro tipo de

dificultades y oportunidades en su desarrollo. Dentro de estas dificultades se encuentra el hecho de que la modalidad del sistema es auditoria y el modo de interacción es mediante el lenguaje hablado.

La interfaz auditoria es aquella que interactúa con el usuario usando puramente sonido, de manera que responde ya sea con voz o sin ésta utilizando *earcons*. Estos últimos corresponden con audios no verbales, comúnmente además de lo anterior incluyen música de fondo y otros ruidos ambientales. Estas interfaces presentan desafíos de diseño debido a la temporalidad del audio y a que, en muchos casos no hay una interfaz visual que pueda ofrecer algún tipo de retroalimentación al usuario.

Cohen, Giangola y Balogh (2004) proponen cinco principios clave en el desarrollo de este tipo de interfaces:

- Información del usuario final: Por lo regular, el diseñador de la Interfaz es el peor usuario con el cual se puede probar la usabilidad de un producto debido a que éstos conocen de antemano el funcionamiento de las interfaces que están realizando y no suelen ver las ambigüedades que estas pueden significar para el usuario nuevo. Por ello se necesita recopilar información de los nuevos usuarios para validar y refinar decisiones de diseño que tengan impacto en el desarrollo del producto.
- Necesidades integradas de los usuarios y del negocio: Se debe saber qué es lo que desean ambas partes, ya que no sólo se requiere saber lo que el usuario desea de la aplicación, sino al mismo tiempo debe estar enfocado en el negocio o la tarea que se necesita resolver, por ejemplo, en el caso de una línea de ayuda de una aerolínea se requiere que el usuario necesite información de diversos servicios como información de vuelos, cancelación, etcétera y que el usuario no divague en el sistema sin realizar ninguna tarea en específico, para esto se requiere que la aplicación guie al usuario a seguir el flujo predefinido para alguna acción, no obstante, también se requiere que el usuario encuentre simple el uso del sistema para no perderse y que tenga metas específicas con las cuales se pueda ayudar del software.
- Definición del sistema desde las etapas iniciales para evitar cambios en las etapas posteriores: En la industria de software los años de experiencia han enseñado a los *Project Managers* las maneras más eficientes de hacer el

deployment de un producto de software de manera eficaz. Para esto se requiere un análisis funcional de alto nivel antes de que los desarrolladores se sumerjan a fondo en el desarrollo de módulos específicos de software. Lo mismo aplica para las VUI ya que el conocimiento del usuario y del negocio representan una ventaja esencial, puesto que este conocimiento reduce el riesgo de realizar decisiones pobres en el diseño. De esta manera disminuye su costo y tiempo de desarrollo debido a los imprevistos causados por malas decisiones en las etapas iniciales. Así mismo, probar la aplicación desde fases tempranas de desarrollo puede proporcionar pistas, guías y evitar riesgos. Las definiciones ayudan a darle consistencia al diseño y de esta forma se mejoran inherentemente los puntos clave relativos a la usabilidad del producto.

- Diseño conversacional de la aplicación: Como se mencionó, los diseñadores algunas veces no logran ver las ambigüedades que el producto representa para los usuarios nuevos, estos errores no vistos en su debido tiempo pueden crear problemas para identificar las acciones a realizar. Para solventar esto se deben especificar las decisiones para los casos de inicio, rechazo o inclusive cuando el tiempo de espera es demasiado. Las malas decisiones pueden crear que el leguaje usado sea de cierta manera innatural y que no sea comprensible tanto para el usuario como para el diseñador. Por ello se debe conocer de antemano el flujo de acciones entre el usuario y el sistema.
- Estudio del contexto de aplicación: El diseño considera el contexto de la conversación y el flujo de llamadas, y es que ésta se debe de desarrollar de acuerdo con el tipo de negocio. Se deben de establecer estándares de órdenes cuando el usuario desea realizar una acción. El contexto del lenguaje es importante ya que no se espera que el usuario use un lenguaje informal o algún tipo de jerga muy aislada para esto el contexto del discurso debe considerar que se pueden tener diferentes acciones para una misma tarea (en el caso de que esto ocurra).

Esos puntos clave ayudan al desarrollo de la metodología que a continuación se presenta y la cual consiste en seis fases

- Definición de requerimientos: La meta de la definición de requerimientos es llegar a comprender la aplicación a detalle. Esta meta va más allá de entender las funcionalidades y sus características, también debe considerar a los usuarios de la aplicación y se enriquece con preguntas tales como ¿Qué los motiva a usar el sistema? ¿Qué expectativas tiene el usuario? y ¿Cuáles son los escenarios de uso?, entre otras. Se deben de entender las metas primarias de la aplicación y cómo éstas interfieren en dado caso con otras aplicaciones que están siendo usadas simultáneamente. Entender los requerimientos no sólo ayuda a aprender las características del sistema sino también otras características, además con esta comprensión de los requerimientos se pueden obtener métricas acerca del éxito del sistema para ayudar a mejorar áreas específicas dentro del mismo.
- Diseño de alto nivel: Éste encapsula lo aprendido en la fase de requerimientos de forma concreta y ayuda a formar un marco de referencia para el diseño detallado. Adicionalmente, al realizar estas decisiones de alto nivel antes de entrar en detalle en la solución del problema, se brinda consistencia y unidad, los cuales nunca surgen cuando un desarrollo es llevado sin algún tipo de planificación.
- Diseño detallado: Incluye la especificación del flujo de llamada y todos los *prompts*. En éste se consideran todos los posibles escenarios para la tarea en mente, inclusive en algunos casos proporcionan ayuda e instrucción en caso necesario, además, se prueba la usabilidad del producto de manera iterativa para obtener el mejor resultado posible.
- Desarrollo: Implica la implementación del flujo de llamada como software, todas las interfaces y bases de datos necesarias, así como también todos los servicios web y otras dependencias que interactúan con la aplicación.
- Pruebas: Esta fase se refiere a la prueba de la aplicación antes de pasar a la fase de prueba piloto; para esto se necesita crear una serie de pruebas en cada una de las fases de desarrollo para comprobar que la aplicación se comporte de la manera esperada. Así mismo, dichas pruebas pueden servir para afinar detalles y conseguir valores iniciales para el reconocimiento, así como medir que su rendimiento sea razonable antes de que la aplicación sea puesta para su uso público. Aunado a esto se pueden realizar pruebas de usabilidad, las

cuales ayudan a encontrar problemas que no necesariamente son detectados a simple vista.

- Tuning: Antes de la liberación, se hacen pruebas piloto con un grupo de usuarios reducido en donde se obtienen audios y otra información para mejorar el rendimiento del producto.

2.3. Domótica

Según Ramlee y otros (2013), el término domótica engloba todas aquellas actividades dentro del hogar, con la intención de ayudar y facilitar la vida diaria mediante la automatización de funciones, lo anterior a través la interconexión de dispositivos.

De acuerdo con Kyas (2013), los elementos más notables de la domótica son:

- Dispositivos bajo control: son todos los dispositivos eléctricos y electrónicos que están conectados y controlados por el sistema domótico.
- Sensores y actuadores: los sensores, metafóricamente hablando, son los ojos y oídos de la red conformada por el sistema domótico, de éstos existe una gran variedad que incluye sensores de temperatura, humedad, luz y nivel de líquido, entre otros. Los actuadores son considerados las manos del sistema, éstos son los medios por los cuales el sistema puede realizar las tareas requeridas en el mundo real, dependiendo del tipo de acción a realizar pueden ser mecánicos, eléctricos o electrónicos.
- Redes de control: éstas proveen la conectividad entre los dispositivos bajo control, los sensores y los actuadores junto con otros dispositivos de control remoto. En la actualidad, el número de dispositivos con tecnologías de transmisión inalámbrica permite la construcción de sistemas domóticos con mayor facilidad. La velocidad de transmisión varía dependiendo del protocolo de comunicaciones y de la distancia entre los dispositivos. También se considera su consumo de potencia y su efectividad.
- Controlador: es el sistema computacional que actúa como el cerebro del sistema domótico, su función es recolectar la información a través de sensores u órdenes emitidas por un control remoto. Actúa basado en instrucciones

predefinidas y realiza las acciones que le son indicadas mediante la activación de los actuadores requeridos.

- Control remoto: actúa conectándose al controlador del sistema domótico mediante la red de control y emite las órdenes que el controlador debe procesar y ejecutar.

2.3.1. Wi-Fi

De acuerdo con Gast (2002) el protocolo IEEE 802.11 a/b/g, también conocido como Wi-Fi (*Wireless Fidelity*), es el estándar utilizado para la creación de redes de área local inalámbrica o WLAN (*Wireless Local Area Networks*). Permite a los usuarios navegar en Internet a velocidades de banda ancha mediante la conexión de dispositivos con un punto de acceso (*Access Point*) o cuando se encuentra en modo *ad hoc*. Su uso es favorecido debido a las ventajas de movilidad que este tipo de red presenta en distancias que van desde 6 metros hasta 45 metros dentro de edificios, y distancias superiores en espacios exteriores.

Esta arquitectura consiste en varios componentes que interactúan para crear una WLAN que permite movilidad transparente a las capas superiores. Para esto, la unidad básica en este esquema se llama BSS (*Basic Service Set*), el cual consiste en un conjunto de estaciones fijas o móviles, si una de estas estaciones sale de su BSS, deja de comunicarse con los demás miembros de la BSS.

El protocolo IEEE 802.11 emplea el IBSS (*Independent Basis Service Set*) y el ESS (*Extended Service Set*). Como indica Flickenger (2002), los diferentes esquemas con los cuales se puede configurar la red y su compatibilidad con Ethernet hacen que este tipo de redes sean fáciles de configurar y de usar.

2.3.2. Internet de las cosas

El concepto de internet de las cosas (*Internet of Things* o IoT por sus siglas en inglés), de acuerdo con (Shuang-Hua, 2014), es conectar todos los dispositivos con posibilidad de conexión a la red global; lo cual presenta una visión en la que el Internet se extiende hacia los objetos cotidianos. El término como tal fue popularizado por el Instituto Tecnológico de Massachusetts en 1999, y se comenzó a propagar debido a la implementación de sistemas que incluían la interconexión de dispositivos de

identificación por radiofrecuencia (*Radio Frequency Identification*, RFID por sus siglas en inglés).

El concepto de cosas en una infraestructura de red hace referencia a cualquier dispositivo real o virtual que interviene en los objetos del día a día, y que, en conjunto con los seres humanos, agentes inteligentes y los datos en la red crean un ecosistema en el cual todos tienen un identificador de red único. De esta manera el conjunto de dispositivos está interconectado, permitiendo que los datos de cada uno de los dispositivos se puedan compartir y utilizar para diferentes fines. Gracias al incremento de la interconectividad y la expansión del Internet, este campo ha llevado a tener un auge importante en el desarrollo de sistemas integrales con diversos fines desde automatización de fábricas hasta automatización de hogares, ya que como mencionan Wortmann y Flüchter (2015), los campos de aplicación de este tipo de tecnologías son tan numerosos como diversos, ya que las soluciones IoT se encuentran expandiendo hacia virtualmente todas las áreas de la vida cotidiana.

Una de estas áreas es la Domótica (*Home automation*) en inglés, la cual permite crear sistemas de sensores para las aplicaciones cotidianas, tales como sensores de temperatura, gas, humedad y agua, entre otros. También permite el control de los diversos dispositivos que se encuentran en el hogar, permitiendo de esta manera un control sencillo e integral de las comodidades que se poseen en el hogar.

Existe una subrama de este tipo de aplicaciones del hogar que tiene como función principal ayudar a personas con discapacidades visuales, auditivas o motrices (Domingo, 2012). Para lograrlo, se necesita la integración de tecnologías y servicios a través de una red doméstica que permita incrementar la calidad de vida. Algunos de estos servicios son los controladores de los equipos en la cocina, de iluminación y puertas, de temperatura, de agua y de seguridad. Así mismo, la información recabada debe poder ser procesada para dar un mejor servicio al usuario mediante alarmas y activación de actuadores que permitan en dado caso manejar una situación de emergencia.

Otra de las características que han dado un auge a este tipo de soluciones es la creciente capacidad de los dispositivos para administrar el consumo de energía, de su conexión a redes, su capacidad de procesamiento y la capacidad del almacenamiento de datos en dispositivos de pequeña dimensión, permitiendo la digitalización de las

funciones clave y la mejora en la capacidad de los productos de la era industrial (Yoo, Henfridsson y Lyytinen, 2010).

2.3.3. Protocolo de comunicaciones MQTT

El protocolo de comunicación MQTT o *Message Queueing Telemetry Transport* se emplea comúnmente en Internet de las cosas y en el paradigma de comunicación máquina a máquina (M2M, *machine to machine*). Este protocolo hace uso directo de los protocolos TCP/IP (Hillar, 2017). Otra característica es que es un protocolo basado en un bróker ligero con la propiedad de mensajería publish/subscribe, la cual tiene un código de implementación pequeño y que requiere poco espacio de almacenamiento, esto permite su ejecución en dispositivos de 8-bits y controladores de 256KB. Además de esto utiliza un ancho de banda pequeño, bajo consumo de energía y trabaja en redes con disponibilidad variable, alta latencia y ancho de banda reducido. Cuenta con la capacidad de negociación de garantía de entrega del contenido.

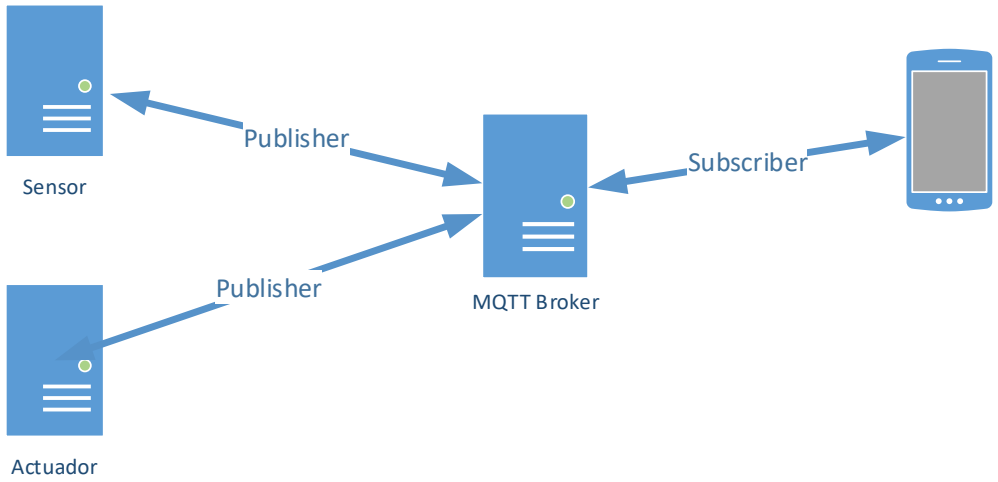


Figura 2.15. Ejemplo de implementación del protocolo de comunicaciones MQTT.

Los sensores o actuadores envían los datos a los dispositivos de monitoreo, dichos sensores o actuadores son considerados como *publisher* y por tanto se encargan de enviar contenidos a los demás dispositivos, mediante el uso de un servidor central llamado MQTT broker, que se encarga de obtener y catalogar cada uno de los mensajes recibidos, reenviar los contenidos recibidos a sus correspondientes destinatarios o *subscribers*. Los *subscribers* son los dispositivos que están interesados en el contenido que envían los *publishers*, de este modo, obtienen la información requerida de una manera sencilla de entender y con un uso de ancho de banda mínimo.

Aparte del MQTT broker son necesarios los clientes MQTT, como tal un cliente MQTT puede ser cualquier dispositivo IoT que reciba y envíe datos de telemetría, lo cual incluye desde un microcontrolador hasta un servidor. Existen dos tipos de clientes en MQTT: *publishers* y *subscribers.*

Para dejar de obtener la información de los *publishers*, el *subscriber* cancela su suscripción al *publisher*. Una ventaja crucial en este tipo de enfoque es el hecho de que los dispositivos *subscribers* y *publishers* no se necesitan conocer de antemano, es decir, un dispositivo tal como un sensor de temperatura puede poseer un número de *subscribers,* pero éste ignorará cuántos sean o cuántas veces hayan sido enviados sus contenidos a los demás dispositivos de esta manera el *publisher* solo se enfoca en publicar la información que le es ordenada.

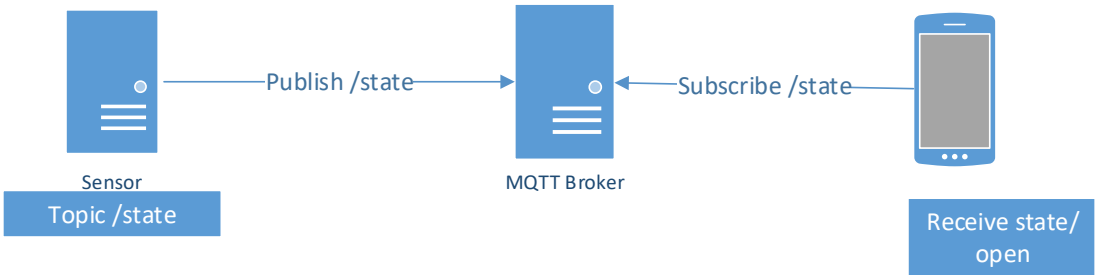


Figura 2.16. Ejemplo de uso de Publisher y Subscriber.

Este protocolo es útil cuando se poseen dispositivos con recursos limitados y, a su vez, este permite: el ahorro de energía, uso en redes con un ancho de banda bajo y donde la conexión no es fiable. Internet de las cosas (IoT) requiere que la conexión de dispositivos ocupe un ancho de banda pequeño y que pueda ser ejecutado en dispositivos de bajos recursos, lo cual en general cumple el protocolo de comunicaciones MQTT.

El tipo de cliente que se use va a depender de su rol en el sistema. Para ello el *publisher* hace uso de temas (*topics)*, los cuales se usan para distinguir el tipo de mensaje enviado. Un ejemplo de esto es suponer que se tiene un sensor de temperatura y otro de humedad. El *publisher* puede declarar que el *topic* de cada uno es "temp/XX" y "hum/XX" respectivamente donde XX es algún valor en cierto tiempo.

El *subscriber,* por otra parte, se debe subscribir al *topic* del *publisher* y con este simple paso de registro de tema se obtendrá periódicamente la información necesaria del tema que se desea.

2.3.4. Raspberry Pi

La Raspberry Pi Foundation (2017) define la plataforma Raspberry Pi como “una computadora de tamaño reducido que es capaz de realizar lo mismo que uno esperaría de una computadora de escritorio, además de esto, tiene la capacidad de interactuar con el mundo exterior, lo que permite crear desde estaciones de clima hasta rockolas”. Calaza (2016) menciona que este sistema sigue la línea de dispositivos que son conocidos como SBC (*Single Board Computer*) debido a su tamaño reducido. En su diseño se utiliza un SoC (*System on a Chip*) que incluye en un sólo chip: procesador, memoria RAM y tarjeta gráfica, para esto vease la Figura 2.17.

Figura 2.17. Raspberry Pi 2 modelo B+.

Este dispositivo tiene varias especificaciones que varian del modelo que se seleccione y actualmente es usada para aplicaciones que requieren mayor capacidad de procesamiento respecto a tarjetas de desarrollo basadas en microprocesadores. Así mismo, debido a su bajo precio presenta una buena opción para realizar soluciones electrónicas.

Edwards (2013) indica que sus aplicaciones van desde su uso como computadoras de bajo costo para firmas de contabilidad y en el desarrollo de sistemas que incluyen Z-Wave, hasta sistemas de automatizacion para compañias como Ciseco y reproductores de medios digitales. Debido a su versatilidad, esta plataforma presenta una excelente alternativa para la creacion de sistemas electrónicos en general.

Capítulo 3. Desarrollo

La solución propuesta en este trabajo requiere de un modelado de alto nivel, a partir del cual se crean las funcionalidades del sistema, lo cual presenta una mejor practica de diseño contra enfoques donde los requerimientos no son controlados desde una fase inicial de desarrollo, ya que la funcionalidad del sistema está basada en una comunicación hablada.

Siguiendo la metodología de desarrollo (Véase Figura 1.6), una vez que se concreta el planteamiento de la VUI se continua con la implementación en hardware y la configuración de los dispositivos. Una vez implementadas todas las partes de este proyecto se procede a integrarlas para obtener el producto que se ha planteado como resultado de este trabajo de tesis.

3.1. Diseño de Alto Nivel

En esta fase se describe la arquitectura de alto nivel para el desarrollo de la VUI. El requerimiento principal consiste en la identificación de palabras y la formación de órdenes usando palabras aisladas mediante un dispositivo móvil. Para esto, se requiere que las órdenes sean capturadas mediante el micrófono del dispositivo móvil y validadas de acuerdo con ciertas reglas. Una vez validadas las órdenes, éstas serán enviadas mediante conexión Wi-Fi a un dispositivo que controla y administra las conexiones y atiende las órdenes enviadas. Como condiciones necesarias se requiere que sea sencillo de usar para cualquier usuario y que trabaje sin conexión a internet, solamente usando una red WLAN.

El propósito principal es proveer una guía al desarrollador para la selección de vocabulario, selección de órdenes y la implementación de los algoritmos. Con esto se puede verificar que todas las funcionalidades necesarias sean cubiertas en este proyecto, para construir eficazmente la aplicación.

3.1.1. Objetivo de la VUI

El objetivo es crear una aplicación llamada Speech Automated Domotics (SAD) la cual se ejecuta sobre el sistema operativo Android. Una vez ejecutada la aplicación, ésta entra en modo de espera, es decir, grabará periódicamente de su entrada de audio

los sonidos del lugar donde se encuentre esperando que el usuario emita alguna de las palabras clave iniciales para desencadenar el flujo de construcción de la orden y así proseguir hasta completar la composición o cancelar el proceso. Para esto, el audio grabado será tomado como una secuencia temporal que no se almacena en el dispositivo. Así mismo, estas órdenes indican una acción en los dispositivos distribuidos a lo largo de toda la casa y por tanto la verificación de órdenes debe formar parte integral del desarrollo.

Una vez que la aplicación completa el proceso e identifica las combinaciones de palabras emitidas como válidas, envía un mensaje a un servidor basado en IoT para ejecutar las operaciones que le son requeridas.

La Figura 3.1 muestra los componentes y la arquitectura de la aplicación basada en su interacción principal con el usuario y las características principales que darán soporte a esta solución.

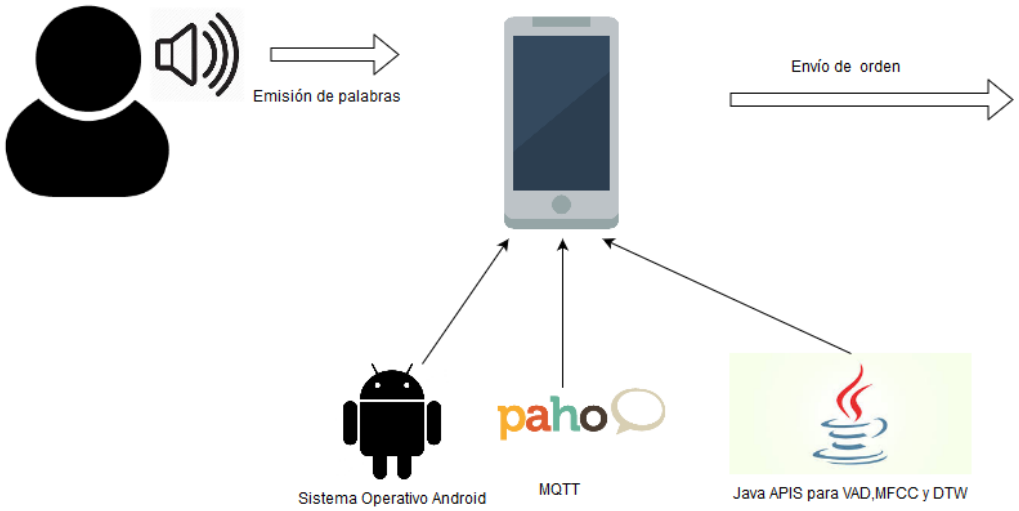


Figura 3.1. Diagrama de la VUI.

3.1.1.1. Perspectiva del producto

El sistema SAD está dividido en varios componentes modulares para su implementación, en su mayoría, éstos fueron programados debido a que no se tiene alguna biblioteca disponible al momento, debido a que éstas se incluyen en toolkits externos o en aplicaciones con licencias propietarias, en el caso de las bibliotecas para el manejo de las comunicaciones su implementación será de código abierto.

El lenguaje de programación principal del proyecto es Java debido a que se adecua de manera eficiente a implementaciones en el sistema operativo Android, y se ejecuta de manera nativa en esta plataforma, permite la ejecución de la aplicación en tiempo real y además existen bibliotecas disponibles.

Para la manipulación de los audios grabados por los usuarios se utiliza la herramienta Audacity que es de código abierto y permite entre otras cosas recortar y eliminar elementos de las señales de audio, esta herramienta se requirió cuando señales de audio presentaban algún tipo de error al ser grabadas. Para la extracción de características se implementó una herramienta basada en Java la cual hace uso de las bibliotecas implementadas (VAD, MFCC, DTW), esta herramienta se ejecuta en cualquier plataforma que tenga instalado JRE en su versión 1.8 como mínimo.

3.1.1.2. Herramientas utilizadas para el desarrollo

El entorno de desarrollo se ejecuta en el sistema operativo Windows 10, como requisito se necesita Java Development Kit (JDK) 1.8 como mínimo. Las herramientas utilizadas son las siguientes:

- Visio 2013: Para el diseño de los diagramas de flujo y herramientas visuales, en su versión de prueba.
- Mindjet MindManager 2013: Para la lluvia de ideas y demás actividades complementarias, en su versión de prueba.
- Paho MQTT: Para la comunicación inalámbrica entre la aplicación y el broker encargado de la administración de actuadores y la red.
- Java API for DTW: Para la ejecución del algoritmo *Dynamic Time Warping*.
- Java API for MFCC: Para la ejecución del *Voice Activity Detection* y la extracción de características mediante MFCC.
- Android Studio: Para la programación de la aplicación Android.
- NetBeans IDE: Para la programación de las API de Java
- MATLAB: Para verificar el funcionamiento de la implementación de algunos algoritmos de procesamiento de señales, en su versión de prueba.
- fluidUI.com: Para el prototipado de las vistas de la aplicación y su flujo.
- Audacity: Para la manipulación de audio.
- VNC Viewer: Para la configuración de los dispositivos y el *broker*.

3.1.1.3. Restricciones generales

SAD es una aplicación fácil de usar, que permite al usuario retroalimentar de manera correcta la información recabada. Así mismo, evita reconfiguraciones innecesarias una vez que ésta sea configurada de la manera correcta, para esto, almacena la configuración. Por otra parte, debe trabajar eficientemente evitando la carga innecesaria del CPU del dispositivo y trabajar sin conexión a internet.

El número de dispositivos a controlar en esta implementación será de 6, para esto se tendrán en cuenta 3 ventiladores y 3 lámparas, cada uno de los cuales se identifica con algún lugar de la casa como puede ser la cocina, el cuarto y la sala. Con base en esto, el vocabulario será restringido con la intención de ser adecuado a las necesidades de la aplicación.

3.1.1.4. Suposiciones

Se asume que el usuario posee un dispositivo Android 7.0 con un procesador quad-core como mínimo y que se ha obtenido un conjunto de entrenamiento de este mismo. Se asume, también que las fallas durante el proceso de implementación se reduzcan con la creación de este documento que da pie a la especificación del producto.

3.1.1.5. Aspectos especiales de diseño

Este proyecto funciona solamente usando una red WLAN sin conexión a internet, además, como es conocido de los protocolos TCP/ UDP, se desconoce con precisión el retardo bajo el cual funciona este sistema. El conjunto de entrenamiento estará empaquetado junto con la aplicación, por lo tanto, ésta es monousuario y requiere de un conjunto de grabaciones realizadas por el usuario antes de ser compilada, empaquetada y ejecutada.

3.1.2. Arquitectura del sistema

Esta fase detalla la arquitectura de alto nivel y las interacciones del sistema. Puesto que esta implementación es un paso inicial se tiene en mente la interacción posterior con el resto del sistema basado en las especificaciones iniciales propuestas a continuación del diseño de la VUI. Así mismo son necesarias la creación e inclusión de

diversas partes del diseño de la interfaz, se requiere un punto de apoyo con el cual las posteriores etapas de comunicación y de configuración de dispositivos se acoplen sin tener que realizar una retroalimentación extensa, ya que esto se traduce en tiempos más largos de implementación y errores inesperados.

El proyecto hace uso de un modo de entrada e interacción, el cual consiste en el micrófono del dispositivo con sistema operativo Android. Para esto, el sistema está atento a la presencia de sonido y todo aquello que presente características similares a la voz será grabado por un intervalo de aproximadamente 2 segundos.

La Figura 3.2. muestra el diagrama de flujo del escenario principal de la aplicación teniendo en cuenta solo las llamadas principales.

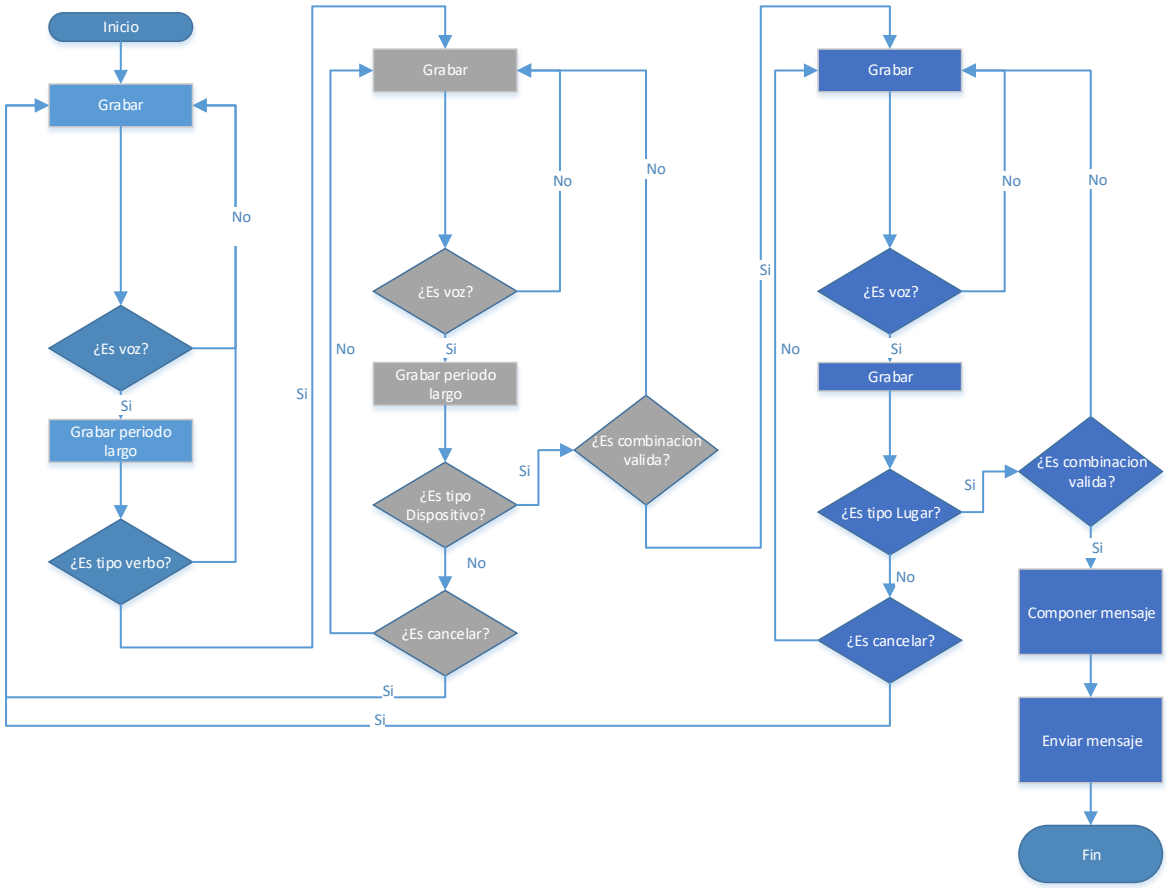


Figura 3.2. Diagrama de flujo del sistema.

3.1.3. Vocabulario

Se requiere que el vocabulario sea fácil de entender y aplicar, debido a las limitaciones que se tienen para la identificación de sentencias completas; también se

debe considerar que las órdenes sean lo más parecidas a las oraciones producidas naturalmente por los humanos, por ello se tienen órdenes basadas en el siguiente esquema:

- Verbo: Indica un conjunto reducido de verbos de las acciones a realizar.
- Dispositivo: Indica el dispositivo a manipular.
- Lugar: Indica el lugar donde se va a ejecutar cierta acción con los dispositivos.

Figura 3.3. Gramática fundamental de las órdenes.

Esas oraciones son comunes como, por ejemplo. “Encender luz del cuarto”, “Encender música de la sala”, “Apagar televisión de la sala”, etc. Por tanto, se considera que el esquema planteado tiene una similitud con el lenguaje natural del ser humano.

En la Tabla 3.1 se listan las palabras clasificadas por su respectivo grupo.

Tabla 3.1. Vocabulario de la aplicación.

Verbo	**Dispositivo**	**Lugar**
Encender	Ventilador	Sala
Apagar	Luz	Afuera
Abrir	Puerta	Cocina
Cerrar		Cuarto
		Enfrente

Aparte de las palabras anteriores, se incluye la palabra **Cancelar**, la cual termina la composición de la orden y regresa la aplicación a un modo de espera. Este tipo de organización permite la expansión en el tipo de órdenes y el tipo de dispositivos a usar. Se ha evitado la declaración estática de órdenes completas y en su lugar se han escogido palabras independientes, con la finalidad de que el tamaño de la base de datos disminuya y se requiera un menor espacio de almacenamiento en el dispositivo. También con la intención de que esto incrementará la tasa de reconocimiento.

El desencadenador válido consiste en el verbo, el cual inicializa el proceso de reconocimiento de órdenes, en la manera posible se ignora la presencia de otras palabras permitiendo que ésta sea el punto de partida de ejecución del algoritmo presentado en el diagrama de flujo de la Figura 3.2.

El usuario tendrá realimentación por parte de la pantalla que permanecerá encendida en todo momento, así mismo, el algoritmo verifica que la secuencia de palabras sea correcta puesto que existen combinaciones de palabras que carecen de sentido, por ejemplo “Encender puerta cocina”, “Encender encender encender” o “Apagar encender puerta”.

3.2. Desarrollo de Bibliotecas

En el diseño de alto nivel se hace uso de un par de bibliotecas basadas en lenguaje Java. A continuación, se describe la implementación del algoritmo para comprender el funcionamiento general de cada una de las partes integrales que dan soporte a la VUI y que permiten el reconocimiento de órdenes. Las bibliotecas propuestas contienen los algoritmos de MFCC, VAD y DTW, respectivamente.

También se presentan los diagramas de flujo que describen el funcionamiento de los algoritmos a nivel general, en el Apéndice C se incluyen los algoritmos en forma legible.

3.2.1. MFCC

Para implementar el algoritmo de extracción de características, se necesitan otros algoritmos para el procesamiento del audio. Por consiguiente, se programaron las siguientes utilerías basado en lo tratado en capítulos anteriores:

- Generación de bancos de filtros triangulares.
- Conversión de Hertz a Mel y viceversa.
- Implementación de ventana Hamming.
- Implementación de la FFT.
- Aplicación de filtros triangulares.
- Transformada Discreta del Coseno.

Por otra parte, se implementó el método de HTK MFCC debido a que presenta un buen rendimiento y a que requiere menos pasos que el método MFCC Slaney, ahorrando tiempo de procesamiento. Pese a que la biblioteca es configurable, ésta quedó limitada debido a que no se implementó la FFT para n-puntos, prefiriéndose la variante de Cooley-Tukey que en este caso limita los puntos a tratar a 2^n. Así mismo, cabe señalar que esta implementación, a diferencia de la HTK MFCC estándar que por defecto usa

24 filtros, se eligió el uso de 40 filtros triangulares, combinando una característica presente en la implementación de MFCC Slaney, esta decisión se llevó a cabo debido a que se consiguieron mejores resultados tal como se puede observar en la Tabla 3.2.

Para esto en la tabla se muestran el efecto de agregar 40 filtros en lugar de los 24 que están por default así mismo se muestra el resultado de agregar los valores de delta y delta-deltas.

Tabla 3.2. Resultados del porcentaje de reconocimiento cambiando el número de filtros y agregando deltas y delta-deltas.

Número de filtros triangulares	Sin deltas	Con deltas
20	64.38%	65.33%
24	69.86%	71.24%
40	73.97%	74.49%
45	71.23%	68.49%

La Figura 3.4 muestra el diagrama de flujo para la obtención de los MFCC, siguiendo las especificaciones iniciales y el método propuesto en el marco teórico (Véase Párrafo 2.1.3.2.2).

3.2.2. VAD

El algoritmo VAD fue diseñado específicamente para este proyecto teniendo en cuenta un funcionamiento simple sin demasiados parámetros configurables, con la intención de que la calibración de este algoritmo sea más sencilla mediante pruebas de usuario. Se eligió un detector basado en energía, como se describe en el Inciso 2.1.1.2. Se consideraron tres características principales: coeficiente de autocorrelación en Lag 1, suma de energía del segmento de audio y número de cruces por cero.

Este algoritmo exige valores de umbral TE (umbral de energía), TCorr (umbral de correlación) y TX (umbral de cruces por cero) para ser ejecutado. Dando de esta manera, mayor flexibilidad a la implementación, así mismo, hace uso de la función para calcular la FFT.

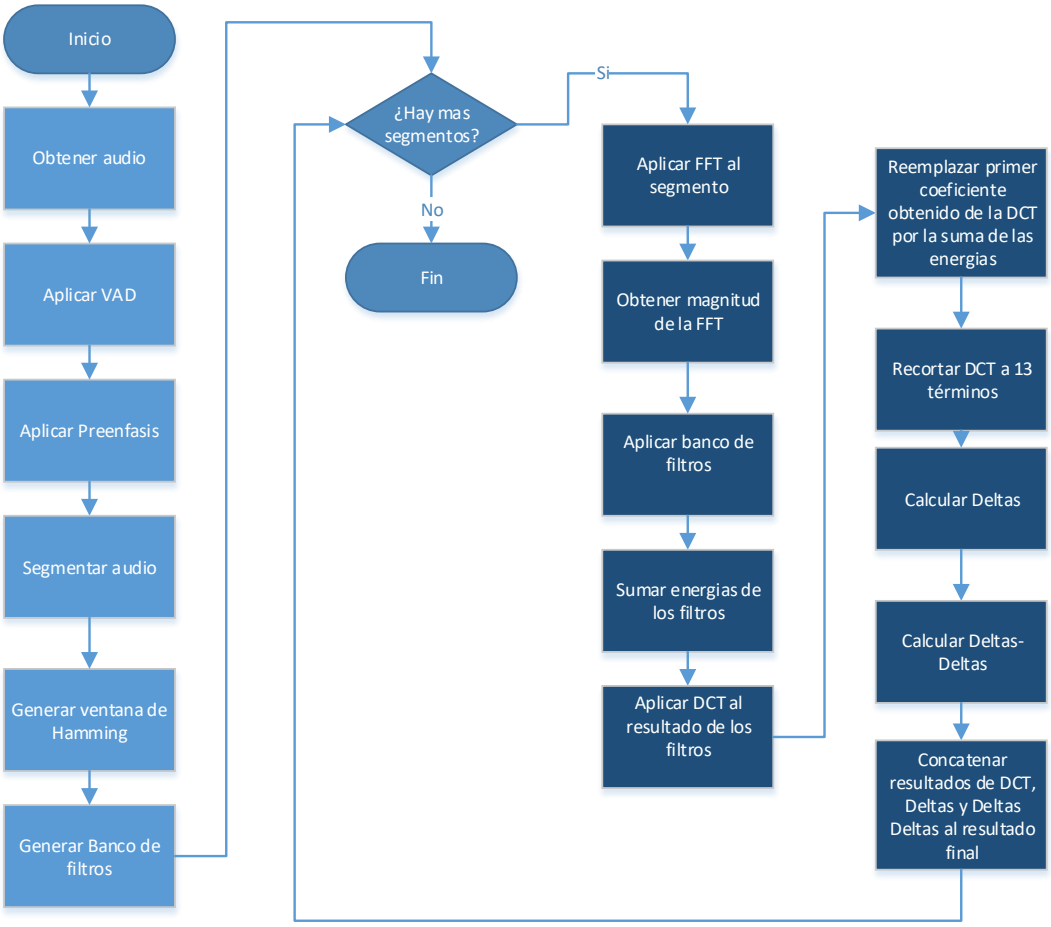


Figura 3.4. Diagrama de flujo de la implementación de los MFCC.

Cabe señalar que se probaron otras medidas para la discriminación de la voz. tales como: media geométrica, media y frecuencia dominante en la FFT. De estas medidas se obtuvo una mejora nula o casi imperceptible, por lo tanto, se desecharon estas ideas a favor de las anteriores, para hacer que el algoritmo se ejecute de manera rápida. La Figura 3.5 presenta el diagrama de flujo del VAD diseñado con las características de coeficiente de autocorrelación en Lag 1, suma de energía del segmento de audio y número de cruces por cero como principales discriminantes.

3.2.3. Distorsión dinámica del tiempo (DTW)

Existen una gran variedad importante de configuraciones para este algoritmo, de las cuales se implementaron varias configuraciones con los resultados mostrados en la Tabla 3.3.

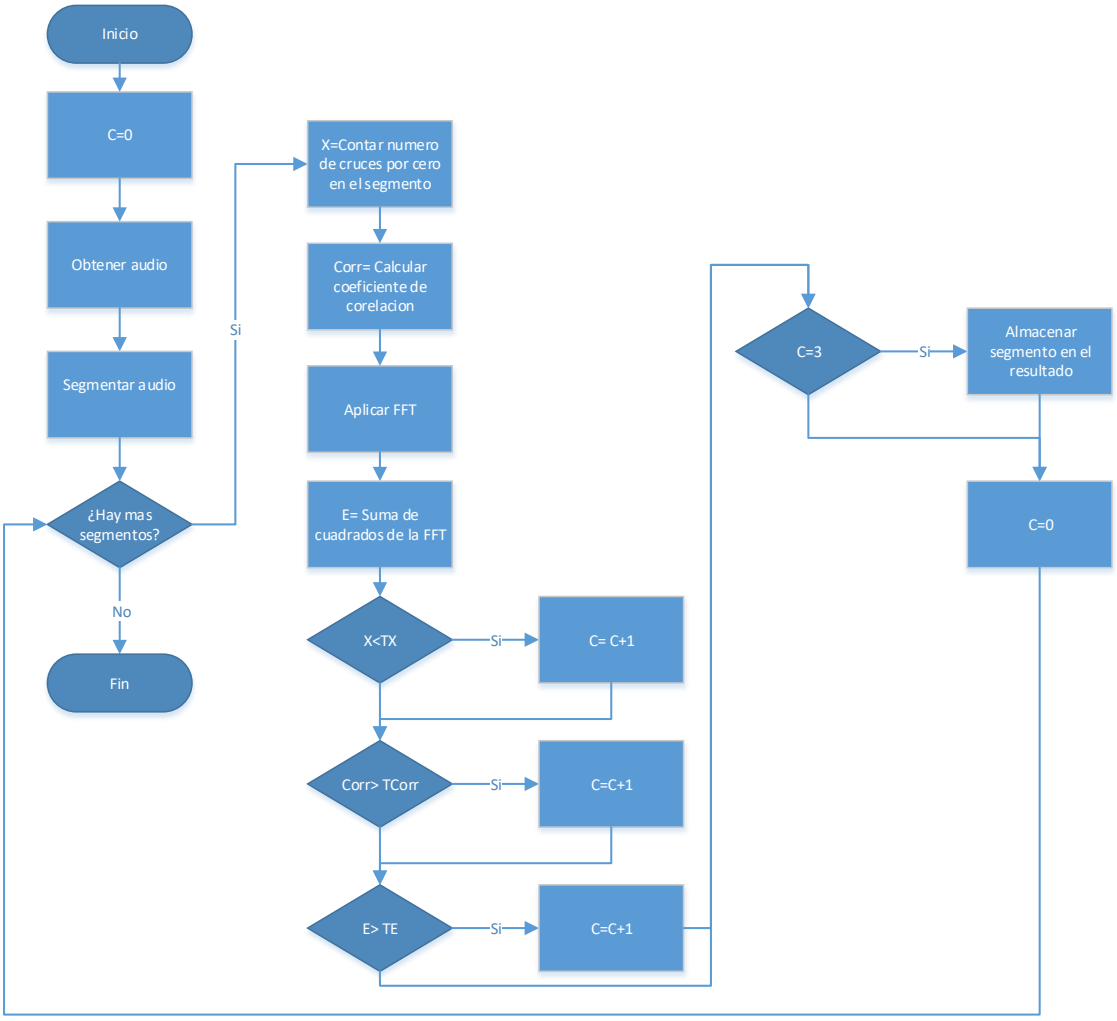


Figura 3.5. Diagrama de flujo del VAD.

Tabla 3.3. Resultados de diferentes configuraciones del algoritmo DTW.

Tipo de Escalón	Ventana de Paliwal con limite recomendado	Sin ventana	Banda Sakoe-Chiba
White y Neely	69.86%	57.53%	71.232%
Sakoe-Chiba Asimétrica P=0	24.65%	35.61%	32.87%
Sakoe-Chiba Simétrica P=0	45.2%	35.61%	56.16%
Sakoe- Chiba Asimétrica P=1	38.35%	56.16%	8.2%
Sakoe- Chiba Simétrica P=1	58.9%	57.53%	12.16%

Se seleccionó una configuración que tiene como restricción de escalón la variación de White y Neely con una ventana Sakoe-Chiba, ya que obtuvo mejores tasas de reconocimiento, así mismo se hace notar que los escalones asimétricos obtienen resultados bajos al aplicar la restricción global. Esto debido a que al ser asimétrica en ocasiones se pueden omitir elementos que son útiles y por tanto presenta perdidas al

aplicar restricciones globales. Se hace notar que la implementación de la restricción de Itakura se omitió debido a que el tiempo de procesamiento de la ventana resultó demasiado largo para la aplicación. La Figura 3.6 muestra el diagrama de flujo del algoritmo implementado.

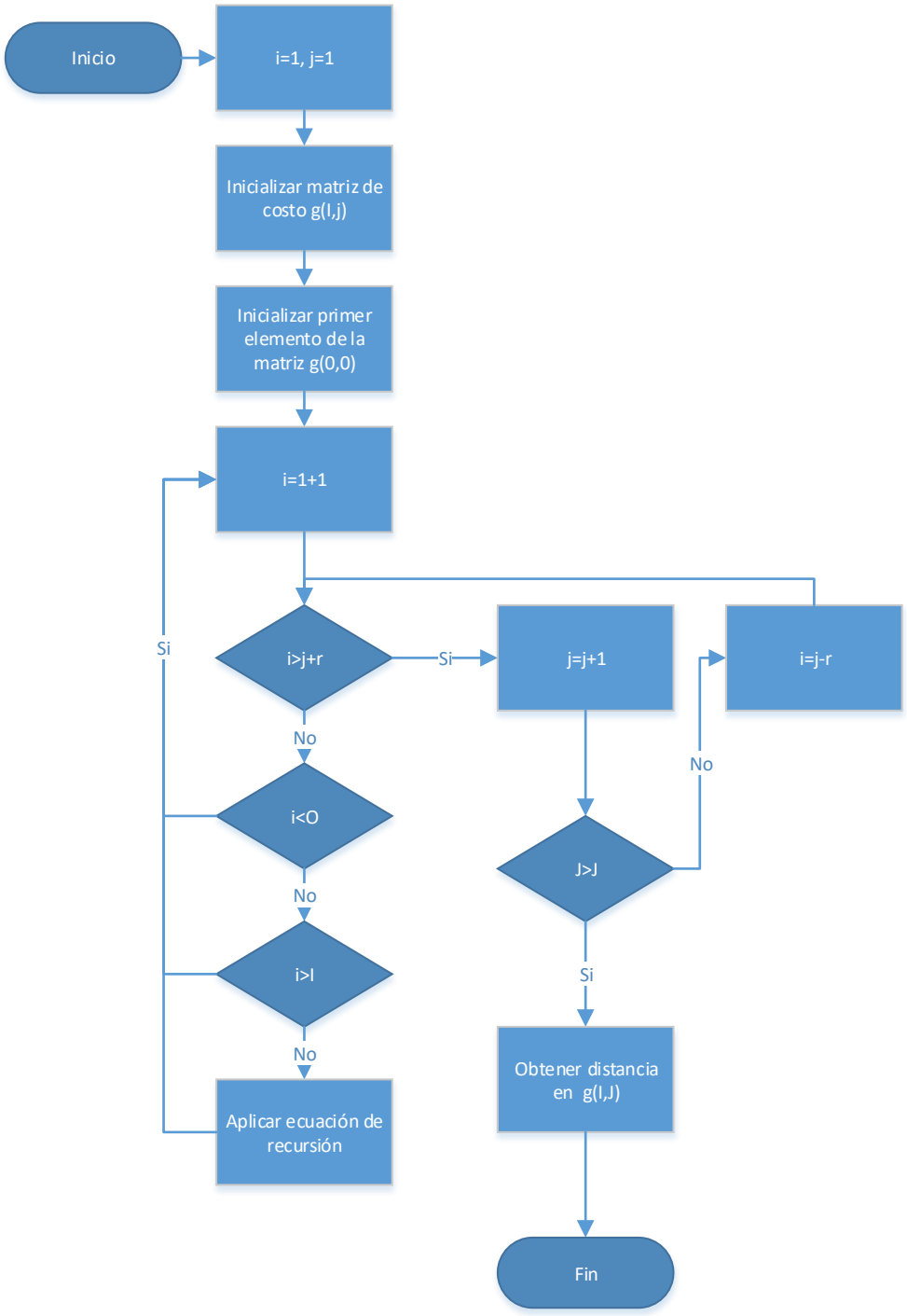


Figura 3.6. Diagrama de flujo del algoritmo DTW.

Para inicializar la matriz local de costos se implementó un método que se basa en inicialización por infinitos en lugar de suma de constantes ya que estas sumas toman tiempo de procesamiento. Esta implementación cambia respecto del esquema original descrito en el Apartado 2.1.1 en su forma de implementación, al declarar infinitos en lugar de una suma de constantes en la columna cero y fila cero. Con esta inicialización se permite una ejecución rápida sin perder eficacia y precisión en el resultado.

3.3. Desarrollo de la Interfaz

El siguiente paso en la construcción de esta parte del proyecto consiste en la creación de la aplicación Android llamada SAD, la cual funciona como control del sistema domótico.

Con base en los requerimientos iniciales y en las funcionalidades requeridas, la aplicación se diseñó con una vista principal contiene la interfaz para creación de órdenes y una pantalla de configuración, en donde se configuran valores para mejorar la calidad del algoritmo VAD y del esquema de extracción de características, la cual será modificada para obtener los mejores resultados para el usuario. Cabe mencionar que la vista principal no contiene ningún tipo de control en su interfaz, sólo contiene indicadores a colores que retroalimentan al usuario en cada una de las fases. La segunda pantalla contiene controles activos, no obstante, ésta es manejada una sola vez, cuando el usuario configura el software, y esta configuración podrá realizarse con la ayuda de un técnico o un familiar en caso de que el usuario lo requiera. La Figura 3.7. muestra el prototipo de la interfaz principal.

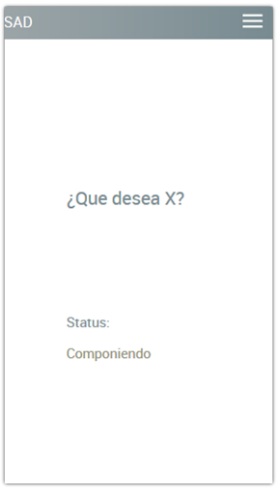


Figura 3.7. Interfaz principal de SAD.

La Figura 3.8 muestra la interfaz de configuración, la cual contiene un par de controles tipo sliders; que definen los parámetros a configurar del algoritmo VAD, del nivel del micrófono y de la tolerancia de ruido en los coeficientes MFCC. Todo esto con la intención de hacer su uso más simple para aquel usuario que intente configurar la aplicación en un punto óptimo.

Por otra parte, estos parámetros son almacenados en la configuración del dispositivo permitiendo que no se necesite configurar de nueva cuenta, a menos que el usuario lo vea necesario.

Figura 3.8. Prototipo de interfaz de la pantalla de configuración.

Como se mencionó, la vista principal mostrada en la Figura 3.7 contiene transiciones, las cuales retroalimentan al usuario el estado actual del programa. Para ello se utilizan cuatro colores para indicar los siguientes estados.

- Verde: Indica que el sistema se encuentra en modo de espera.
- Rojo: Indica que existen errores en la aplicación.
- Amarillo: Indica que la aplicación está generando una orden.
- Azul: La orden ha sido generada y enviada exitosamente.

En una sección especial de la aplicación se muestra el estado en el cual se encuentra el usuario para la correspondiente composición de la orden, de esta manera se propone hacer más entendible el proceso. Se espera que el número de errores sea

nulo, así mismo, el envío de la orden es automático una vez que se complete la composición.

Con la necesidad en mente de que el producto sea reconocible, la Figura 3.9 muestra el icono usado para esta aplicación.

Figura 3.9. Icono de la aplicación.

3.3.1. Implementación de la aplicación móvil

La implementación se llevó a cabo en Android Studio, el cual es un entorno de desarrollo gratuito (IDE). Se siguieron las guías de diseño anteriores propuestas en las secciones 3.1.1, 3.1.2 y 3.3 y después de codificar las funciones necesarias se obtuvieron los siguientes resultados. Las figuras 3.10 y 3.11 muestran las vistas de la aplicación implementada.

La pantalla de configuración quedó con los siguientes parámetros configurables:

- Sensibilidad al ruido en correlación: este parámetro modifica el límite de correlación del algoritmo VAD, con este parámetro se desecha las muestras que no contienen audio, es decir elimina los puntos que solo contienen estática debido al micrófono.
- Nivel de ruido mínimo de activación: Este parámetro define cual es el nivel de sonido mínimo para que la aplicación considere el audio actual para la siguiente fase de procesamiento.

- Nivel mínimo de volumen en audio: Este parámetro sirve para eliminar sonidos que se presenten en el audio pero que no contengan el nivel mínimo para ser considerados en el algoritmo VAD y por lo tanto sean removidos, en este por ejemplo se eliminan pronunciaciones que no tienen el nivel mínimo de volumen requerido.

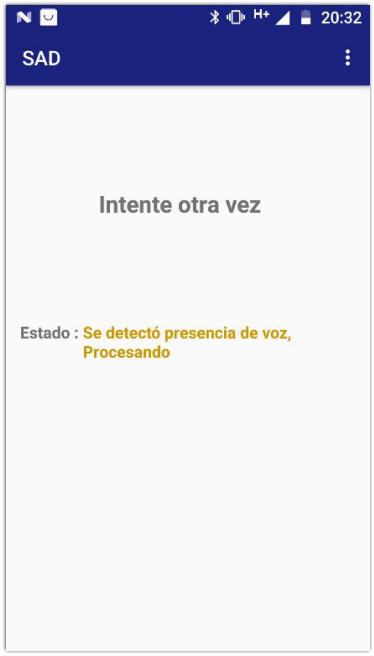


Figura 3.10. Vista principal de la aplicación.

Figura 3.11. Vista de configuración de la aplicación.

Una vez implementada la aplicación se integran todas las partes del sistema y se configuran tanto los dispositivos como la implementación del *broker* con el que la aplicación se comunica de manera inalámbrica.

3.4. Implementación de las Comunicaciones Inalámbricas

La comunicación de la aplicación con la tarjeta es mediante WiFi utilizando un router como puerta de enlace. El *router* seleccionado para esta aplicación es el modelo N301 del fabricante Tenda.

La Figura 3.12 muestra la interconexión de los dispositivos y cómo se comunican entre ellos.

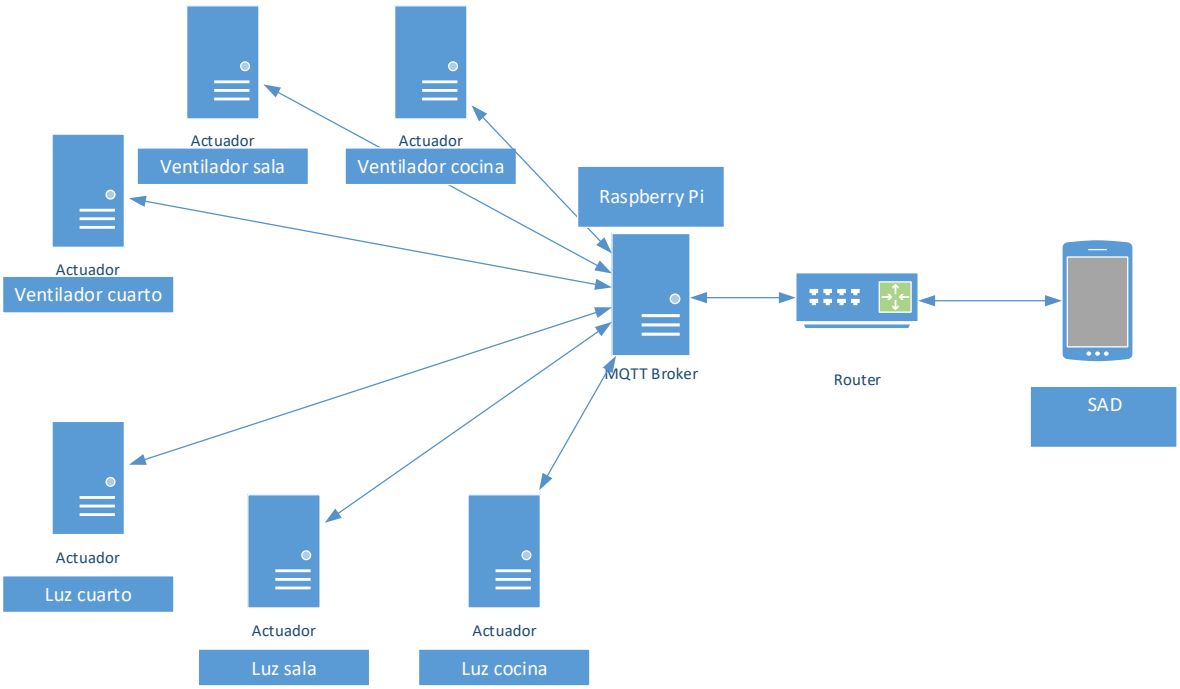


Figura 3.12. Diagrama de comunicaciones inalámbricas.

Considerando el diagrama de comunicación de la Figura 3.12, se procedió a configurar el *router* para que otorgara las direcciones a cada uno de los dispositivos. Como se menciona inicialmente en el apartado 2.3.3, los dispositivos actuadores no necesitan una configuración especial tal como IP estática debido a que el único dispositivo que debe de tener este tipo de IP configurada es el *broker,* en este caso la tarjeta Raspberry Pi. La configuración del router se puede observar en la Figura 3.13.

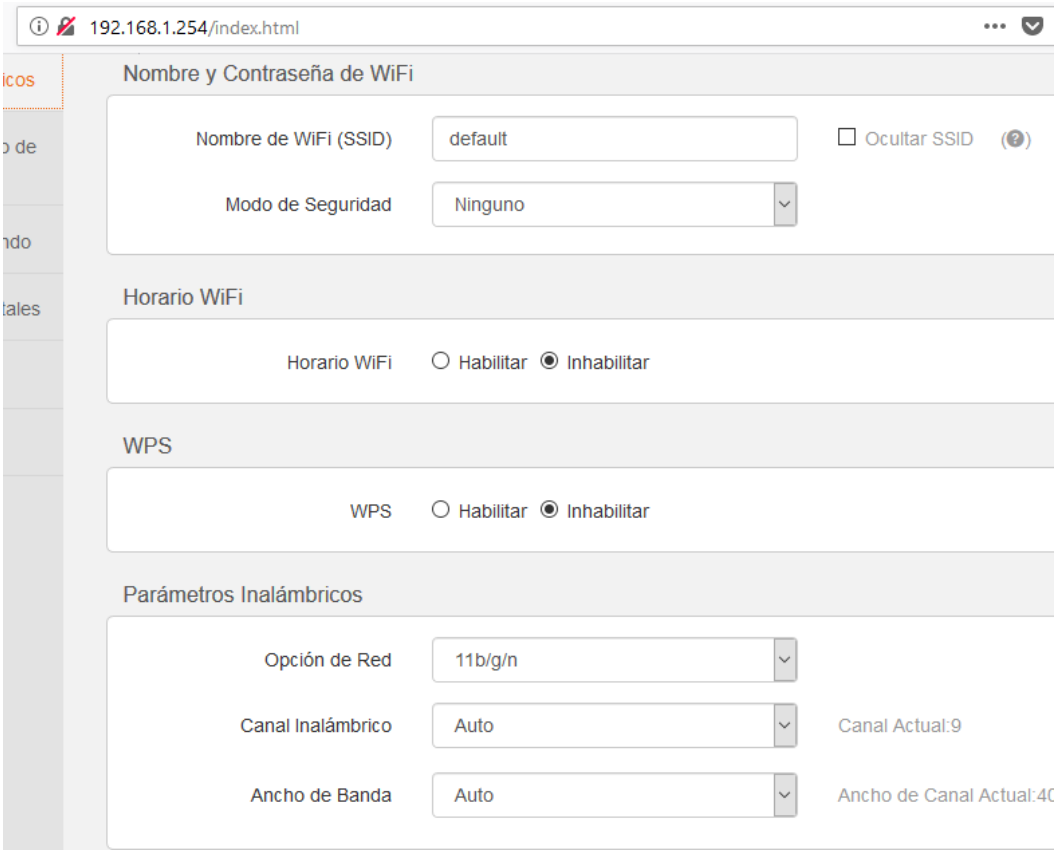


Figura 3.13. Configuración de la red inalámbrica.

Además de esto se estableció el *router* para trabajar con DHCP, se verificó que no hubiera un firewall bloqueando el puerto TCP 1883.

Con este diseño inicial, se procedió a agregar los demás dispositivos restantes a la red. Para el *broker* se modificó la configuración por defecto de una tarjeta Raspberry Pi, se consideraron primariamente: su capacidad de procesamiento de esta misma y el consumo de energía. También, la plataforma Raspberry Pi tiene la capacidad de instalar el software necesario haciendo unas modificaciones básicas a las fuentes de instalación del dispositivo.

El sistema operativo seleccionado para esta plataforma es Raspbian en su versión 8.0. Se instaló Eclipse Mosquitto, el cual es un MQTT *Broker* de código abierto con licencia EPL/EDL que implementa las versiones del protocolo de comunicaciones MQTT en sus versiones 3.1 y 3.1.1. Mosquitto es ligero y es compatible con aplicaciones de bajo consumo de energía y recursos de procesamiento, así mismo se presta para implementar aplicaciones en servidores industriales. Para observar el procedimiento realizado se puede consultar el Apéndice A.

3.5. Configuración de los Actuadores

Se eligió el switch inalámbrico diseñado por ITEAD cuyo modelo es el Sonoff Basic. Este actuador es de fabricación china y permite la actualización de firmware de ESPurna. En la Figura 3.14 se muestra el dispositivo seleccionado.

Figura 3.14. Sonoff Basic.

El firmware de estos dispositivos puede ser modificado por una versión especial de código libre que puede ser ESPurna, Tasmota, etc. que permite su manipulación y configuración para evitar software propietario y agregarles flexibilidad a los dispositivos, además estos incluyen un microcontrolador ESP8266 que cuenta con un módulo Wi-Fi, GPIO's y puede ser programado mediante el entorno de programación Arduino. Por otra parte, se instaló el firmware ESPurna, programado bajo licencia de código abierto que permite configurar varios servicios, entre ellos el protocolo de comunicación MQTT. Para ello se realizó la modificación necesaria de estos dispositivos, para esto se puede consultar el Apéndice B de este documento.

3.6. Resultados de la Configuración de los Dispositivos

Para comprobar la comunicación de todos los dispositivos con el router se requiere que los actuadores y el *broker* hayan quedado correctamente comunicados. En la Figura 3.15 se observa la terminal de Mosquitto con un actuador correctamente conectado, en este caso se trata del ventilador de la sala.

```
521165713: New connection from 192.168.1.104 on port 1883.
521165713: New client connected from 192.168.1.104 as VentSala (c0, k30).
521165713: Sending CONNACK to VentSala (0)
521165714: Received UNSUBSCRIBE from VentSala
521165714:     #
521165714: VentSala #
521165714: Received SUBSCRIBE from VentSala
521165714:     /sonSalaVent/relay/+/set (QoS 0)
521165714: VentSala 0 /sonSalaVent/relay/+/set
```

Figura 3.15. Resultado de la configuración de los actuadores.

Con las configuraciones obtenidas se modifican los valores de IP del *broker* y el puerto TCP que requiere la aplicación móvil para establecer la comunicación, cabe señalar que en este caso el usuario de la aplicación tiene el nombre de AndroidUser como se muestra en la Figura 3.16.

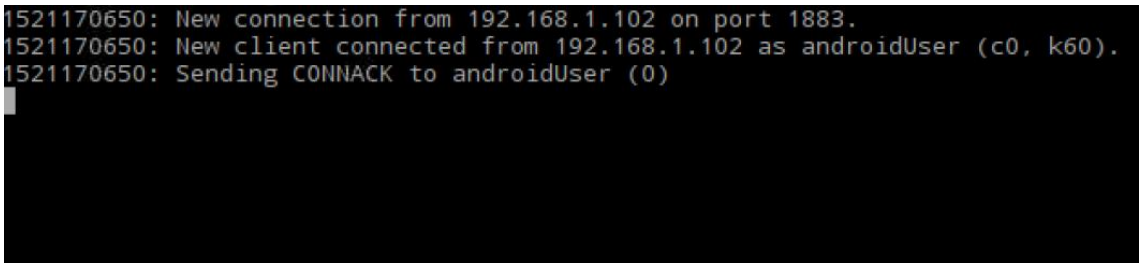


Figura 3.16. Resultado de la configuración de la aplicación móvil.

Una vez establecida la comunicación entre la aplicación móvil, el *broker* y los actuadores, se procedió a realizar pruebas para verificar la funcionalidad del sistema de acuerdo con las especificaciones iniciales.

Capítulo 4. Pruebas y Resultados

Para verificar el correcto funcionamiento del sistema se realizaron pruebas de caja gris y de caja negra. Cabe señalar que no se realizaron pruebas de caja blanca en la totalidad del sistema, debido a que las bibliotecas *third party* utilizadas para la comunicación ya vienen validadas y su funcionamiento se evaluará en las pruebas de caja gris como parte del sistema completo.

4.1. Pruebas de Caja Gris

En esta primera fase de pruebas se examinó el funcionamiento de los algoritmos de procesamiento de señales implementados en Java: FFT, DCT tipo II y Coeficiente de Autocorrelación Lag 1.

Para corroborar el correcto funcionamiento de las implementaciones en Java se compararon los resultados obtenidos con los correspondientes en el software de procesamiento de datos Matlab[1]. Para ello, se generaron señales de prueba sintéticas, las cuales se exportaron de Matlab a Java para examinar que los algoritmos de procesamiento de señales implementados en Java funcionaran de manera correcta.

Estas señales de prueba consisten en señales sinusoidales de diferentes frecuencias. Una de estas consiste en una suma de sinusoidales y la otra en una sinusoidal simple con solo una frecuencia presente como se puede observar en el siguiente apartado.

Debido a la naturaleza de los números de punto flotante y a su expresión, se decidió comparar los resultados de manera manual con ayuda de tablas que contienen los resultados de cada una de las implementaciones, con base en estos resultados se calculan los errores entre las implementaciones correspondientes, haciendo uso de la definición de error absoluto mostrada en la ecuación (26).

$$\Delta x = |x_o - x| \tag{26}$$

donde Δx denota el error absoluto y x_o el resultado obtenido en el algoritmo a probar y x el resultado obtenido por el algoritmo base.

[1] Con esto se asume que los componentes e implementaciones presentadas por Matlab han sido comprobadas y se le han realizado las pruebas necesarias, lo cual significa que la implementación es correcta.

Una vez obtenidas estas tablas de los errores absolutos se procedió a calcular el Error Absoluto Promedio utilizando la ecuación (27).

$$EAP = \frac{1}{n}\sum_{i=1}^{n}|x_i - x_{bi}| \tag{27}$$

donde x_i denota el resultado i-esímo de un algoritmo implementado en Java, x_{bi} denota el resultado i-esímo del algoritmo base y n denota el numero de resultados totales de salida del algoritmo, con esto se obtuvo el error de precisión de los algoritmos implementados. Para que una implementación se considerase como apta se dio como máximo un margen de .001 de error.

Además de lo anterior se utilizaron gráficas que permitieron comparar los resultados de manera visual corroborando que las implementaciones son equivalentes.

Siguiendo con este mismo esquema se probaron las demás implementaciones que son dependientes de estos algoritmos base los cuales consisten en: Banco de filtros triangulares, *Detección de Actividad de Voz* y *Distorsión Dinámica de Tiempo*. Con los resultados de estas pruebas se corrigieron errores en los programas base antes de pasar a la siguiente fase.

En el siguiente apartado se muestran los resultados para cada uno de los algoritmos previamente mencionados, también se procedió a probar parcialmente el sistema de manera que se obtuvo un perfil del funcionamiento del sistema de comunicaciones, así como el tiempo de respuesta de la red y de los dispositivos, para obtener una estadística general y conocer el tiempo de ejecución de las órdenes.

Se realizaron pruebas sobre el tiempo de respuesta de la red utilizando el protocolo TCP y enviando paquetes de 128 bytes para simular la carga de la red en los dispositivos actuadores. Con ello se obtuvieron los tiempos de respuesta promedio de la red. Por último, se procedió a realizar pruebas del tiempo de respuesta de los actuadores usando el protocolo de comunicaciones MQTT para observar el tiempo promedio de respuesta que tendría un usuario.

4.1.1. Resultados de las pruebas de caja gris

Los algoritmos de procesamiento de señales son la base del sistema, en específico en la etapa de extracción de características y, por lo tanto, es importante que se

implementaran siguiendo la metodología (véase Subcapítulo 1.8). Para esto se compararon los resultados de los algoritmos empleando las señales de prueba mostradas en la Figura 4.1.

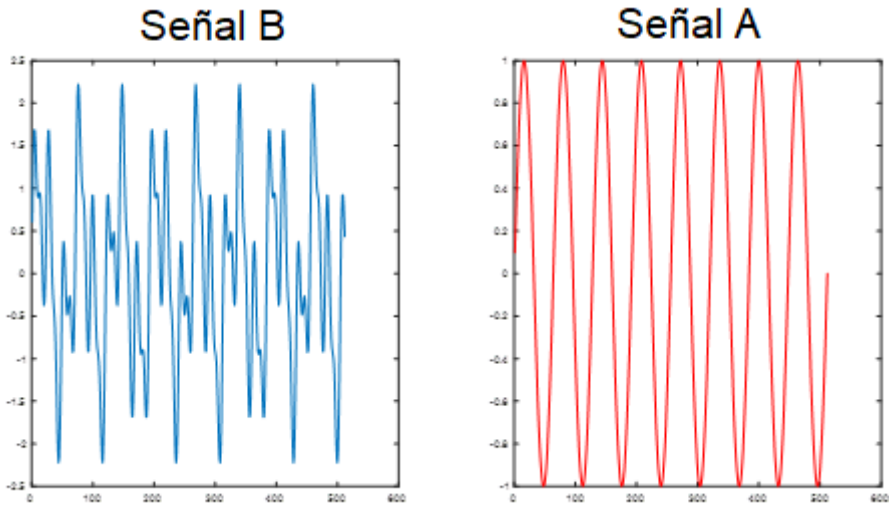


Figura 4.1. Señales de prueba de los algoritmos. La señal roja es una señal simple la cual será referida como A y la señal azul es una señal compleja compuesta de varias componentes a distintas frecuencias la cual será referida como B.

La Tabla 4.1 muestra los resultados de la comparación de la implementación de la FFT realizada y la implementación de Matlab respecto al error promedio en la parte Real, Compleja y finalmente el valor absoluto, este último es el más importante ya que es el utilizado para el procesamiento de las señales de voz.

Tabla 4.1. Resultados de las pruebas sobre el algoritmo FFT.

Señal	**Error promedio Real**	**Error promedio Complejo**	**Error Promedio en Valor absoluto**
A	1.40E-16	4.60E-16	3.59E-15
B	-7.65365E-15	-4E-15	3.39933E-14

Como se observa en la Figura 4.2, los resultados obtenidos y graficados son similares al ser superpuestos.

Siguiendo con las pruebas, se analizaron los resultados obtenidos al aplicar la implementación a la señal de prueba B.

Para corroborar estos resultados, se procede a superponer los resultados y como se puede observar en la Figura 4.3, los resultados obtenidos son similares.

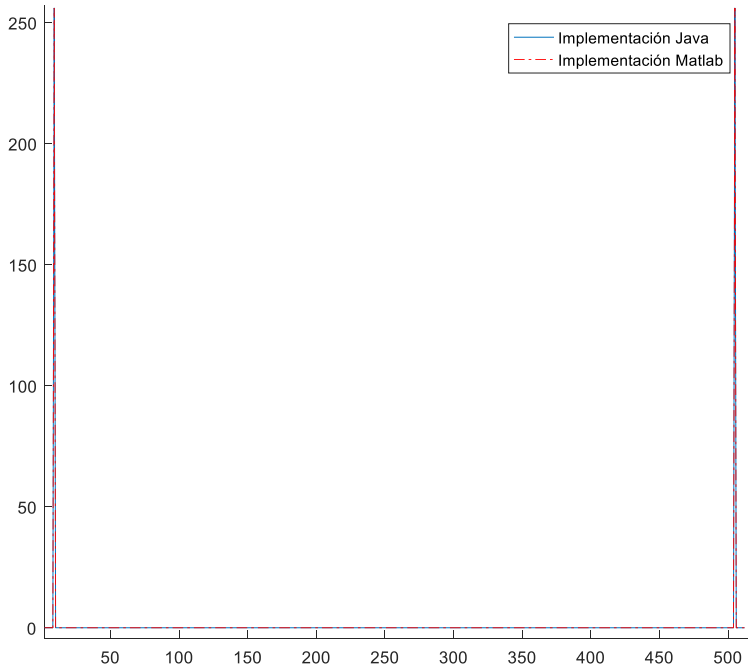


Figura 4.2. Resultados obtenidos de la FFT superpuestos.

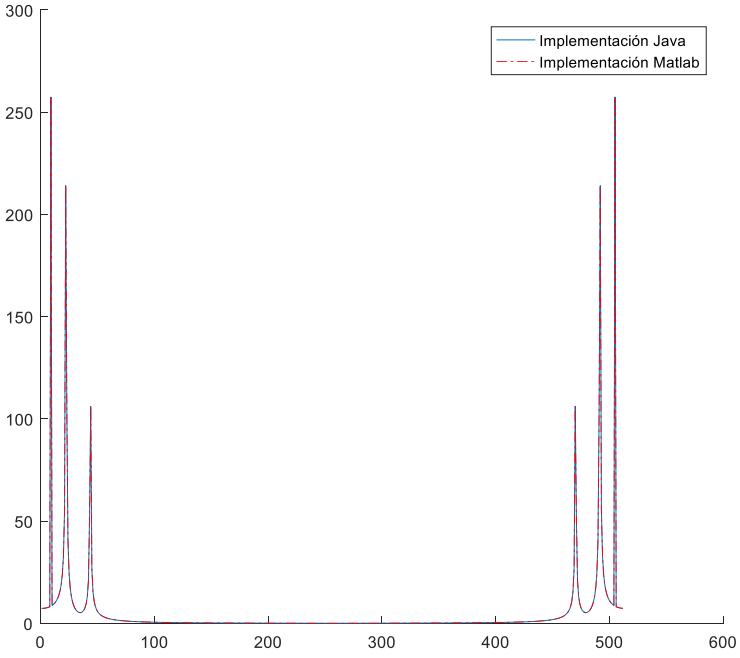


Figura 4.3. Resultados obtenidos de la FFT superpuestos.

Al estudiarse la Tabla 4.1 en conjunto con las Figuras 4.2 y 4.3 puede observarse que el error presente en la implementación es de 1.11E-15 %. Por lo tanto, estos resultados cumplen los requisitos principales de la implementación.

Se continuó con las pruebas de la implementación de la Transformada Discreta del Coseno (DCT), la Tabla 4.2 muestra los coeficientes obtenidos de su procesamiento.

Tabla 4.2. Resultado de las pruebas sobre el algoritmo DCT.

Señal	Error promedio
A	9.74606E-15
B	0

Como se puede observar en la Figura 4.4, existe una similitud entre las gráficas de los resultados del procesamiento de la señal A al ser superpuestas.

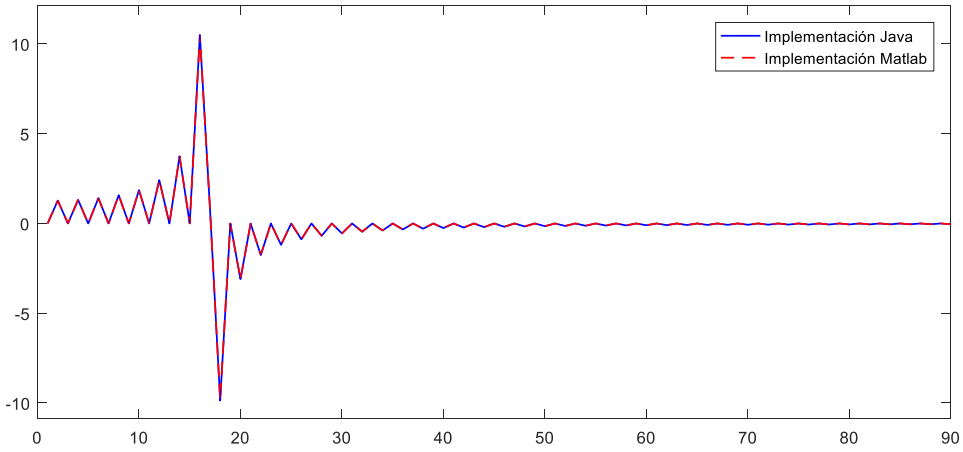


Figura 4.4. Resultados de DCT superpuestos.

Como se puede observar en la Figura 4.5, los resultados superpuestos del procesamiento de la señal B coinciden.

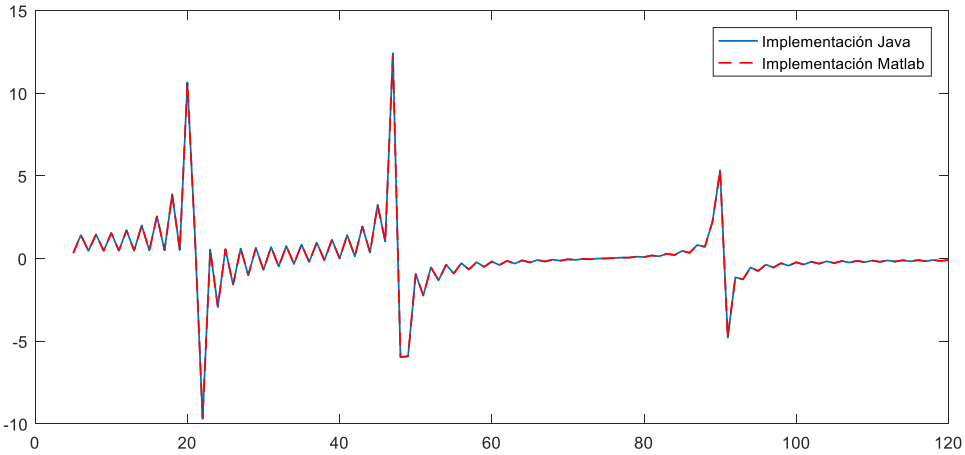


Figura 4.5. Resultados superpuestos de la DCT aplicada a la señal B.

Al examinar los resultados, en la Tabla 4.2 se puede observar que los errores obtenidos numéricamente son de poca significancia, es decir son de 0 y de 9.74606E-15 por lo que se concluye que la implementación cumple con los requerimientos.

En la Tabla 4.3 se muestran los resultados de la prueba del coeficiente de Autocorrelación en Lag 1, el cual se utiliza en el procesamiento del *Voice Activity Detection*.

Tabla 4.3. Resultado de las pruebas del algoritmo de Autocorrelación en Lag 1.

Señal	Implementación Java	Implementación Matlab	Error
A	0.995222076	0.995184727	3.73E-05
B	0.968160297	0.96770286	0.000457

Una vez probados los algoritmos anteriores, se procedió a verificar cada una de las salidas de las utilerías y clases restantes para comprobar que los algoritmos se ejecuten bien, particularmente los bancos de filtros triangulares, la salida del Voice Activity Detection y el algoritmo DTW. La Figura 4.6 presenta los resultados obtenidos de los bancos de filtros triangulares en Java, los cuales se exportaron a Matlab para ser visualizados.

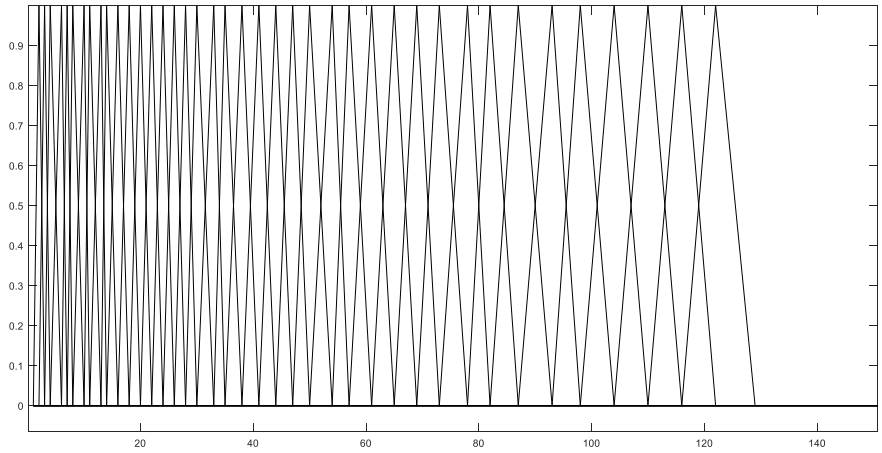


Figura 4.6. Bancos de filtros triangulares.

Como se puede observar en la Figura 4.6, este resultado cumple con lo que se observa en la Figura 2.8 del presente documento, la cual muestra un ejemplo de una implementación de un Banco de filtros triangulares. Se observa que el espaciamiento entre los filtros triangulares es el adecuado, así mismo la amplitud en estos es la que se deseaba.

La Figura 4.7 muestra los resultados de la implementación del *Voice Activity Detection*, con una parte de las muestras del conjunto de entrenamiento.

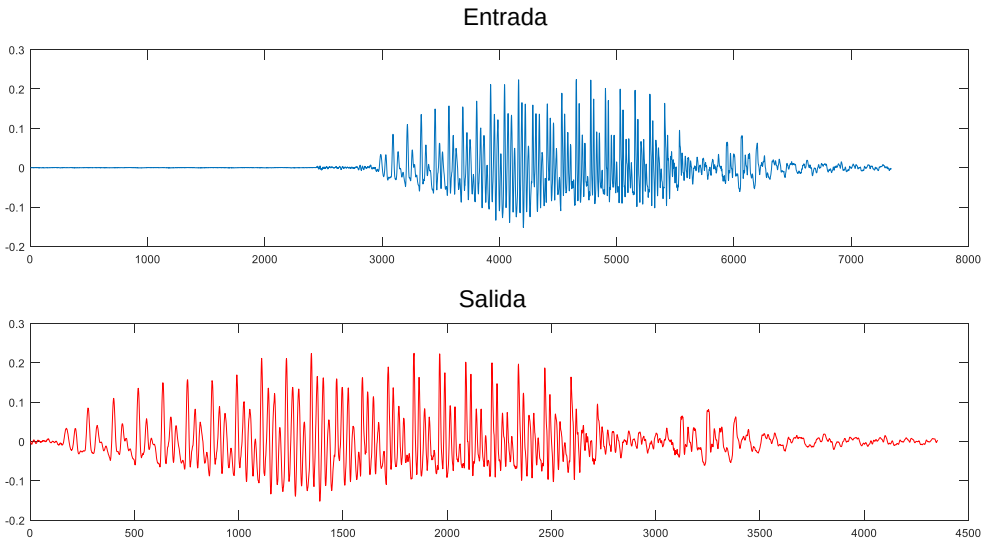


Figura 4.7. Resultado de prueba de VAD.

Para obtener estos resultados se usó una configuración de 150 cruces por cero, energía presente en la señal mayor a 1 y coeficiente de autocorrelación de 0.85. Los resultados de este algoritmo varían dependiendo de la configuración impuesta, no obstante, como se puede observar en la Figura 4.7, los resultados en esta ejecución para la señal de prueba son los siguientes: se redujo su dimensión de 7432 a 4352 elementos lo cual indica una reducción de 40% de elementos. Por lo tanto, el resultado variará dependiendo de la configuración, presentándose casos donde la mejora y reducción es considerable.

El último algoritmo por probar es el DTW, el cual se utiliza para probar la similitud de dos señales de tiempo. Para la prueba de este algoritmo se utilizaron 4 señales de prueba para comparar los resultados con el algoritmo base presentado en Matlab. En la Figura 4.8 se muestran las señales de prueba.

Las señales tienen las siguientes características: La señal A es una onda sinusoidal pura sin algún tipo de ruido, la señal B tiene un ruido blanco, la señal C presenta un menor número de muestras a la señal original A y la señal D presenta ruido blanco con mayor magnitud y además un offset de .5.

Con las señales de prueba se realizaron las siguientes comparaciones de patrones: a) la señal A y B, b) la señal A y C y c) la señal A y D. Una vez obtenidos los resultados, se procedió a compararlos y a obtener el error absoluto obtenido entre los procesamientos de estas señales. De esta manera se obtuvo la Tabla 4.4 que presenta el cálculo del error resultante.

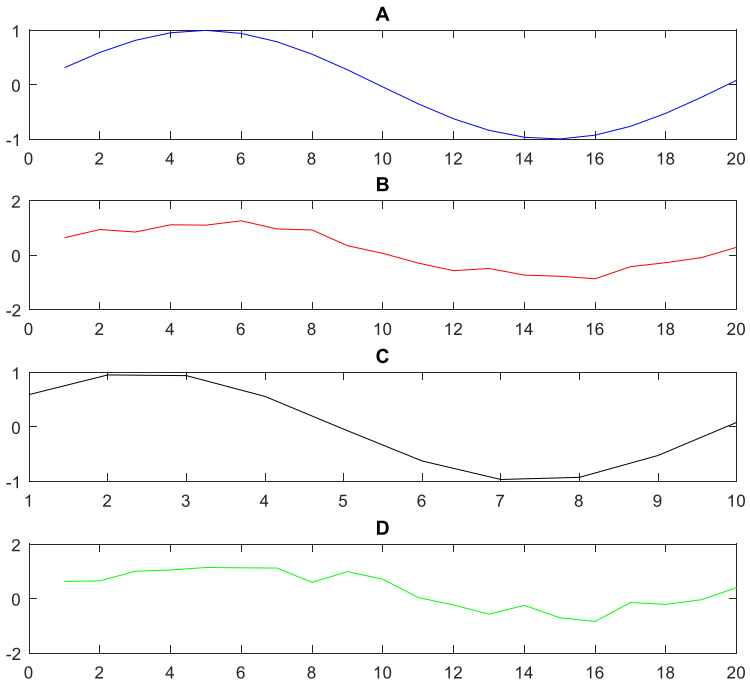


Figura 4.8. Señales de prueba.

Tabla 4.4. Resultados obtenidos de la ejecución de pruebas del algoritmo DTW.

Operación	DTW Java	DTW Matlab	Error
A y B	2.898057	2.898057	5.12E-08
A y C	1.793089	1.793089	7.38E-08
A y D	3.766324	3.766324	3.54E-07

Una vez terminadas las pruebas de las utilerías de clase se procedió a obtener un perfil del tiempo de respuesta de cada uno de los dispositivos y de la red en general, así mismo, se midió el tiempo de respuesta de los actuadores como respuesta a una orden emitida (véanse Figura 4.9 y Tabla 4.5).

Tabla 4.5. Resultados del tiempo de respuesta de los actuadores.

Actuador	Mínimo (s)	Máximo (s)	Promedio (s)
Luz Cuarto	0	11	4.6
Ventilador Cuarto	0	7	2.6
Luz Sala	0	10	3.6
Ventilador Sala	0	7	2
Luz Cocina	0	7	2.6
Ventilador Cocina	0	9	3
Global	0	11	3.06666667

Para realizar el perfil de la red se utilizó la herramienta *ping*, la cual permite observar el tiempo de respuesta de una interfaz cuando recibe un paquete. Para estas

pruebas se utilizaron 500 paquetes de 128 bytes. En la Tabla 4.6 se muestran los resultados conteniendo el mínimo, el máximo y el promedio de tiempos de respuesta para estos dispositivos, así mismo, se muestra el número de paquetes perdidos y el promedio global de los dispositivos.

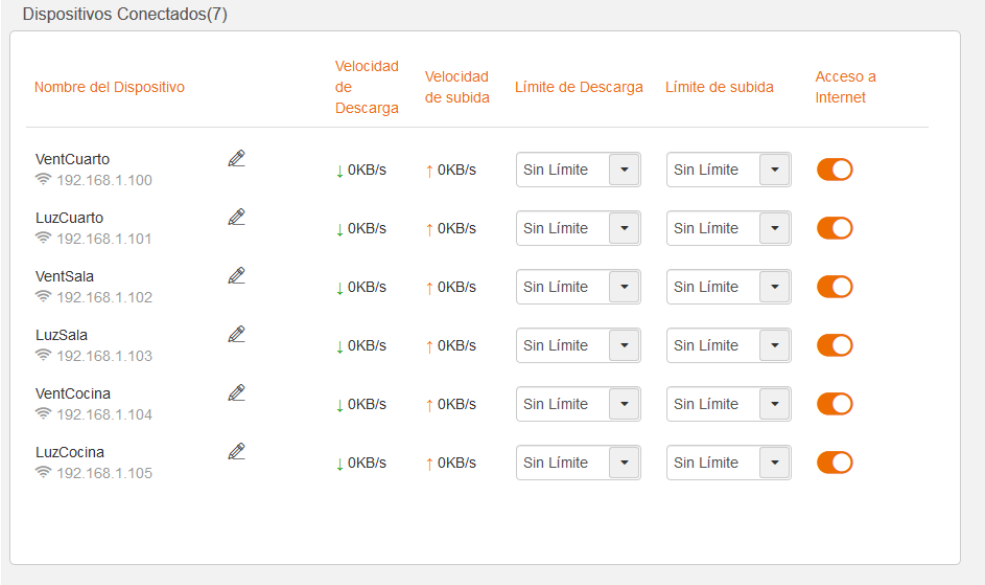


Figura 4.9. Dispositivos utilizados para ejecutar las pruebas.

Tabla 4.6. Resultado de pruebas de respuesta.

Dispositivo (IP)	Mínimo (ms)	Máximo (ms)	Promedio (ms)	Paquetes perdidos
192.168.1.100	2	270	7	0
192.168.1.101	1	148	6	1
192.168.1.102	2	204	6	0
192.168.1.103	1	158	6	1
192.168.1.104	2	180	5	1
192.168.1.105	1	117	6	2
Global	1.5	179.5	6	0.83333333

A continuación, se procedió a obtener el tiempo de respuesta del actuador, es decir, no sólo el tiempo en el que recibe los paquetes sino el tiempo que tarda en activarse una vez que la orden es dada, la Tabla 4.5 muestra los resultados obtenidos. En este caso se emitieron 10 órdenes a cada uno de los actuadores para obtener su comportamiento promedio de los mismos, se obtuvieron los valores mínimos, máximos y promedio en el tiempo de respuesta de activación (véase Tabla 4.5).

4.2. Pruebas de Caja Negra

Para validar el correcto funcionamiento del sistema en su totalidad, se emplearon 4 conjuntos de entrenamiento consistentes en grabaciones realizadas por 4 usuarios, así mismo, estos 4 usuarios probaron el sistema con la finalidad de verificar que el sistema funcionara correctamente. Los conjuntos de entrenamiento debían cumplir con un conjunto de características para obtener la mejor calidad posible. Así mismo, se presentaron las siguientes guías que ayudaron a crear cada conjunto de entrenamiento:

- Cada una de las palabras deberían tener un total de 12 grabaciones (muestras).
- 4 fueron grabadas a una distancia de 25 cm. del hablante, 4 a 50 cm. del hablante y otras 4 mientras el hablante estaba recostado a una distancia de 30 cm.
- Se procesó y verificó la calidad del conjunto obtenido, esto mediante el análisis de los audios mediante el software Audacity.
- Se creó la base de datos para incluirla en la aplicación.

Así para seguir con las pruebas de caja negra se consideraron los siguientes puntos para realizarlas:

- Se tomaron 10 órdenes de prueba aleatorias de las 20 posibles.
- Se administró el conjunto de palabras y de órdenes válidas para permitir al usuario conocer vocabulario y las órdenes a realizar.
- El porcentaje de órdenes correctas del número total determina el rendimiento del sistema.
- Se calculó la precisión total para obtener la precisión del algoritmo de reconocimiento de voz.
- Se cuantificó el número de palabras reconocidas incorrectamente.
- El personal encargado de administrar las pruebas sólo estuvo presente para ver el desarrollo de éstas.

4.2.1. Resultados de las pruebas de caja negra

Una vez que se obtuvieron los conjuntos de entrenamiento se procedió a verificar el funcionamiento del sistema total bajo un esquema en el que 4 sujetos de pruebas

usaron la aplicación. Así mismo, para estas pruebas, 3 de los sujetos de pruebas desconocían completamente la aplicación y cómo usarla. Con esto se tuvo la intención de observar si los usuarios nuevos podían utilizarla sin haber tenido ningún tipo de experiencia con el sistema.

Otro de los puntos importantes fue obtener un porcentaje de efectividad del sistema, para esto se realizaron pruebas teniendo en cuenta 10 órdenes de prueba. Cada una de éstas tendrá como procedimiento obtener cuántas veces las palabras fueron reconocidas de manera correcta y cuántos errores hubo en la fabricación de las órdenes.

Para obtener el resultado de las palabras reconocidas correctamente se usó una matriz de confusión por usuario para permitir obtener resultados cuantificables. Para esto se obtendrá la precisión total la cual se puede calcular con la siguiente ecuación:

$$Precisión\ total = \frac{Clasificación\ total}{Verdad\ total} \tag{28}$$

También cabe mencionar que los cuatro sujetos de pruebas son masculinos, mayores de edad.

El sujeto de prueba 1 realizó las órdenes mostradas en la Tabla 4.7, este usuario realizo pruebas en un dispositivo con las características mostradas en la Tabla 4.8.

A continuación, en la Tabla 4.7 se muestran las órdenes de prueba señalando con una x en la columna Resultado las órdenes incorrectas:

Tabla 4.7. Órdenes de prueba del sujeto de prueba 1

Verbo	**Dispositivo**	**Lugar**	**Resultado**
Abrir	Puerta	Enfrente	
Abrir	Puerta	Cocina	
Apagar	Ventilador	Cuarto	
Cerrar	Puerta	Enfrente	
Apagar	Luz	Cuarto	
Apagar	Luz	Afuera	
Encender	Luz	Sala	x
Apagar	Ventilador	Sala	
Encender	Luz	Cuarto	
Cerrar	Puerta	Cocina	

Tabla 4.8. Características del dispositivo de prueba del sujeto de prueba 1.

Sistema Operativo	Android 7.0 (Nougat)
Chipset	Qualcomm MSM8953 Snapdragon 625
Procesador	Octa-core 2.0 GHz Cortex-A53
Memoria RAM	2 GB

Con lo anterior se obtuvo un resultado de 90% de órdenes correctas. Con la información recabada de las órdenes se obtuvo la siguiente matriz de confusión mostrada en la Tabla 4.9.

Tabla 4.9. Matriz de confusión obtenida del sujeto de prueba 1.

	Encender	Ventilador	Sala	Apagar	Luz	Afuera	Abrir	Puerta	Cocina	Cerrar	Cuarto	Enfrente	Cancelar	Clasificación total	Precisión producida
Encender	2													2	1
Ventilador		2												2	1
Sala			1											1	1
Apagar				4										4	1
Luz					4									4	1
Afuera						1								1	1
Abrir							2							2	1
Puerta								4						4	1
Cocina									2					2	1
Cerrar										2				2	1
Cuarto			1								3			3	0.75
Enfrente												2		2	1
Cancelar													3	3	1
Verdad total	2	2	2	4	4	1	2	4	2	2	3	2	3	96.96	
Precisión	1	1	0.5	1	1	1	1	1	1	1	1	1	0		

Con los datos obtenidos de la Tabla 4.9 se calculó la precisión total la cual resulto de 96.96%. Con ello solo una palabra resultó ser mal clasificada. A continuación, se muestran los resultados del sujeto de prueba 2. Éste hizo uso de un dispositivo con las características mostradas en la Tabla 4.10.

Tabla 4.10. Características del dispositivo del sujeto de prueba 2.

Sistema Operativo	Android 8.0 (Oreo);
Chipset	Qualcomm MSM8953 Snapdragon 625
Procesador	Octa-core 2.0 GHz Cortex-A53
Memoria RAM	4 GB

En la Tabla 4.11.se presentan las órdenes que realizó el sujeto de prueba 2.

Con lo anterior se obtuvo un resultado de 90% de órdenes correctas. Con la información recabada de las órdenes se obtuvo la siguiente matriz de confusión mostrada en la Tabla 4.12.

Tabla 4.11. Órdenes de prueba del sujeto de prueba 2.

Verbo	**Dispositivo**	**Lugar**	**Resultado**
Cerrar	Puerta	Cocina	
Apagar	Luz	Sala	
Encender	Ventilador	Cocina	
Apagar	Luz	Cuarto	
Encender	Luz	Afuera	
Apagar	Ventilador	Cocina	
Apagar	Ventilador	Sala	x
Cerrar	Puerta	Enfrente	
Encender	Ventilador	Sala	
Abrir	Puerta	Cocina	

Tabla 4.12. Matriz de confusión obtenida del sujeto de prueba 2.

	Encender	Ventilador	Sala	Apagar	Luz	Afuera	Abrir	Puerta	Cocina	Cerrar	Cuarto	Enfrente	Cancelar	Clasificación total	Precisión producida
Encender	3													3	1
Ventilador		4												4	1
Sala			1											1	1
Apagar				4										4	1
Luz					3									3	1
Afuera						1								1	1
Abrir							1							1	1
Puerta								2						2	1
Cocina									3					3	1
Cerrar										2				2	1
Cuarto			2								1			1	0.33
Enfrente												1		1	1
Cancelar													3	3	1
Verdad total	3	4	3	4	3	1	1	2	3	2	1	1	3	93.5	
Precisión	1	1	0.33	1	1	1	1	1	1	1	1	1	0		

Con los datos obtenidos de la Tabla 4.12 se calculó la precisión total la cual resulto de 93.5%. Con ello dos palabras fueron mal clasificadas.

A continuación, se muestran los resultados del sujeto de prueba 3. Este hizo uso de un dispositivo con las características mostradas en la Tabla 4.13. En la Tabla 4.14 se presentan las órdenes que realizó el sujeto de prueba 3.

Tabla 4.13. Características del dispositivo del sujeto de prueba 3.

Sistema Operativo	Android 7.0 (Nougat);
Chipset	Exynos 8890 Octa
Procesador	Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)
Memoria RAM	4 GB

Tabla 4.14. Órdenes de prueba del sujeto de prueba 3.

Verbo	**Dispositivo**	**Lugar**	**Resultado**
Apagar	Luz	Cocina	
Cerrar	Puerta	Enfrente	
Abrir	Puerta	Enfrente	
Encender	Ventilador	Cocina	
Apagar	Luz	Cuarto	
Apagar	Luz	Sala	x
Cerrar	Puerta	Cuarto	
Encender	Ventilador	Cuarto	
Encender	Luz	Afuera	
Encender	Ventilador	Sala	

Con lo anterior se obtuvo un resultado de 90% de órdenes correctas. Con la información recabada de las órdenes se obtuvo la siguiente matriz de confusión mostrada en la Tabla 4.15.

Tabla 4.15. Matriz de confusión obtenida del sujeto de prueba 3.

	Encender	Ventilador	Sala	Apagar	Luz	Afuera	Abrir	Puerta	Cocina	Cerrar	Cuarto	Enfrente	Cancelar	Clasificación total	Precisión producida
Encender	4													4	1
Ventilador		3												3	1
Sala			1											1	1
Apagar				3										3	1
Luz					4									4	1
Afuera						1								1	1
Abrir							1							1	1
Puerta								2						2	1
Cocina									3					3	1
Cerrar										2				2	1
Cuarto			1								3			3	0.75
Enfrente												2		2	1
Cancelar													4	4	1
Verdad total	4	3	2	3	4	1	1	2	3	2	3	2	4	97.05	
Precisión	1	1	0.5	1	1	1	1	1	1	1	1	1	0		

Con los datos obtenidos de la Tabla 4.15 se calculó la precisión total la cual resulto de 97.05%. Con ello solo una palabra resulto mal clasificada.

A continuación, se muestran los resultados del sujeto de prueba 4. Este hizo uso de un dispositivo con las características mostradas en la Tabla 4.13. En la Tabla 4.17 se presentan las órdenes que realizó el sujeto de prueba 4.

Tabla 4.16. Características del dispositivo del sujeto de prueba 4.

Sistema Operativo	Android 8.1 (Oreo);
Chipset	Qualcomm MSM8916 Snapdragon 410
Procesador	Quad-core 1.2 GHz Cortex-A53
Memoria RAM	1 GB

Tabla 4.17. Órdenes de prueba del sujeto de prueba 4.

Verbo	**Dispositivo**	**Lugar**	**Resultado**
Cerrar	Puerta	Enfrente	
Abrir	Puerta	Enfrente	
Abrir	Puerta	Cuarto	
Apagar	Luz	Sala	
Encender	Luz	Cocina	
Encender	Ventilador	Sala	
Apagar	Ventilador	Sala	
Encender	Luz	Sala	
Abrir	Puerta	Cocina	
Encender	Luz	Cuarto	

Con lo anterior se obtuvo un resultado de 100% de órdenes correctas. Con la información recabada de las órdenes se obtuvo la siguiente matriz de confusión mostrada en la Tabla 4.18.

Tabla 4.18. Matriz de confusión obtenida del sujeto de prueba 4.

	Encender	Ventilador	Sala	Apagar	Luz	Afuera	Abrir	Puerta	Cocina	Cerrar	Cuarto	Enfrente	Cancelar	Clasificación total	Precisión producida
Encender	4													4	1
Ventilador		2												2	1
Sala			4											4	1
Apagar				3										3	1
Luz					5									5	1
Afuera						1								1	1
Abrir							3							3	1
Puerta								3						3	1
Cocina									2					2	1
Cerrar										1				1	1
Cuarto											2			2	1
Enfrente												2		2	1
Cancelar													4	4	1
Verdad total	4	2	4	3	5	1	3	3	2	1	2	2	4	100	
Precisión	1	1	1	1	1	1	1	1	1	1	1	1	0		

Con los datos obtenidos de la Tabla 4.18 se calculó la precisión total la cual resulto de 100%.

En la Tabla 4.19 se muestra el resultado general de la clasificación de las palabras aisladas.

Como se puede observar el resultado general de precisión total es de 97.03%. En el caso de órdenes completadas correctamente se obtuvo un porcentaje de 92.5%.

Tabla 4.19. Matriz de confusión con los resultados obtenidos.

	Encender	Ventilador	Sala	Apagar	Luz	Afuera	Abrir	Puerta	Cocina	Cerrar	Cuarto	Enfrente	Cancelar	Clasificación total	Precisión producida
Encender	14	0	0	0	0	0	0	0	0	0	0	0	0	14	1
Ventilador	0	11	0	0	0	0	0	0	0	0	0	0	0	11	1
Sala	0	0	7	0	0	0	0	0	0	0	0	0	0	7	1
Apagar	0	0	0	13	0	0	0	0	0	0	0	0	0	13	1
Luz	0	0	0	0	16	0	0	0	0	0	0	0	0	16	1
Afuera	0	0	0	0	0	4	0	0	0	0	0	0	0	4	1
Abrir	0	0	0	0	0	0	7	0	0	0	0	0	0	7	1
Puerta	0	0	0	0	0	0	0	12	0	0	0	0	0	12	1
Cocina	0	0	0	0	0	0	0	0	10	0	0	0	0	10	1
Cerrar	0	0	0	0	0	0	0	0	0	7	0	0	0	7	1
Cuarto	0	0	4	0	0	0	0	0	0	0	9	0	0	9	0.69
Enfrente	0	0	0	0	0	0	0	0	0	0	0	7	0	7	1
Cancelar	0	0	0	0	0	0	0	0	0	0	0	0	14	14	1
Verdad total	14	11	11	13	16	4	7	12	10	7	9	7	14	97.03	
Precisión	1	1	0.63	1	1	1	1	1	1	1	1	1	1		

Capítulo 5. Conclusiones

La presente tesis tuvo como objetivo diseñar e implementar un sistema domótico controlado por órdenes de voz basado en una tarjeta de desarrollo Raspberry Pi mediante órdenes de voz desde un dispositivo móvil con sistema operativo Android.

Esto implicó la necesidad de utilizar un enfoque basado en VUI y para esto se propuso el uso de una metodología hibrida VUI/Hardware. En lo que respecta al reconocimiento de voz, se observó que el desarrollo en el lenguaje Java permitió una rápida migración del proyecto ya que este trabajo se prototipó primero usando un entorno de desarrollo para una computadora personal para después ser migrada a un entorno de desarrollo para dispositivos móviles. Por tanto, el primer prototipo desarrollado en Java fue implementado en una computadora personal, esto con el fin de verificar que los algoritmos implementados funcionaran correctamente. Una vez realizada esta parte, la migración a los dispositivos móviles ocurrió sin contratiempos debido a que no se tuvo que reprogramar código de manera extensiva, lo cual como se pudo observar presentó una ventaja y menor perdida en el tiempo de desarrollo.

El seguir la metodología de VUI permitió a su vez que la implementación de las interfaces fuera concisa ya que desde la creación del diseño de alto nivel se tuvo conocimiento de lo que se esperaba del sistema y así mismo facilitó el desarrollo de las interfaces gráficas eliminando de estas todos los componentes activos que no eran requeridos tales como botones, sliders, etc.

Otro punto principal de esto consistió en la elección de la restricción de escalón y restricciones globales del algoritmo DTW, de esto se obtuvo que la restricción de White & Neely presento mejores tasas de reconocimiento que otros esquemas como los escalones de Sakoe-Chiba. También se observó que la obtención de deltas y delta-deltas en los MFCC produjo un mejor resultado en el proceso de extracción de características. Con esto se obtuvo en general una tasa de reconocimiento del 97.03%.

Por otra parte, se diseñó el esquema de comunicaciones del sistema, este se basó principalmente en el protocolo MQTT ya que permite intercambiar mensajes entre dispositivos con un uso bajo de los recursos de red y de energía. Así mismo mediante este esquema de comunicaciones se consiguió la compatibilidad con los dispositivos móviles y con el lenguaje de programación usado sin dificultad alguna.

Al comparar el precio de los actuadores con protocolo MQTT resultó más económico con respecto al precio de los actuadores basados en Z-Wave por tanto se obtuvo un sistema relativamente económico respecto a otras soluciones.

Con las convenciones propuestas se obtuvo una configuración general del protocolo que incluye los dispositivos actuadores, el dispositivo móvil y el *broker*. Los dispositivos Sonoff Basic de ITEAD, mediante una modificación hardware, permitieron instalar una versión de firmware de código abierto llamada ESPurna que soporta y permite la configuración del protocolo de comunicaciones MQTT. Además de esto el mismo firmware ESPurna en los actuadores permitió realizar copias de respaldo de las configuraciones permitiendo que las estas pudieran ser reconfiguradas en caso de pérdida sin requerir ningún tipo de configuración extra o reconfiguración, lo cual presenta una ventaja en caso de eventos inesperados.

Cabe señalar que los principales problemas que surgieron a la hora de configurar los dispositivos fue un retardo aleatorio en las comunicaciones, debido principalmente a la programación interna de los dispositivos actuadores y sobre la cual no se tiene control. Pese a ello, nunca se interrumpió la comunicación entre los dispositivos, ni se presentaron errores al momento de ejecutar las órdenes deseadas.

Así pues, al tomar en cuenta todo lo realizado, la integración de las partes del sistema se llevó a cabo sin ningún problema ya que al haber establecido un rol y una configuración para cada uno de los dispositivos presentes en la red se pudo observar que el sistema de comunicaciones funcionó correctamente.

Al realizar las pruebas se utilizaron varios dispositivos, los cuales presentaban la característica de tener al menos procesadores con 4 núcleos, 1 GB de memoria RAM y sistema operativo Android 7.0 o superior. Debido a que la aplicación permitía la configuración especial de cada uno de los dispositivos, la variedad de hardware que se usó no presento problemas permitiendo que de cada uno de los diferentes dispositivos móviles obtuvieran un porcentaje de reconocimiento alto, resultando en una aplicación confiable y estable.

Se observó que en ninguno de los casos se produjeron errores de compatibilidad por tanto no se necesitó modificar el código para que la aplicación tuviera compatibilidad con las diferentes versiones de sistema operativo usadas, dando de esto una aplicación de control estable. Esto se puede observar al analizar que del envió de las órdenes

realizadas se obtuviera un porcentaje de efectividad de 90% en general en todos los usuarios.

Con lo anterior se cumplieron los objetivos de este trabajo de tesis y con ello se valida la hipótesis, ya que fue posible diseñar e implementar un sistema domótico basado en una tarjeta de desarrollo Raspberry Pi controlada inalámbricamente mediante órdenes de voz desde un dispositivo móvil con sistema operativo Android.

5.1. Trabajos Futuros

Debido a la restricción de tiempo que se tuvo en la realización de este trabajo quedan algunas tareas que pueden aumentar la importancia de este trabajo. A continuación, mencionare las más importantes.

En el desarrollo de este trabajo se pudo observar la importancia del algoritmo VAD el cual produjo mejores resultados en la clasificación de las palabras para esto se ha de notar que el esquema usado basado en medidas de energía es un esquema ya poco usado, en este caso se implementó debido al corto tiempo de desarrollo. Por tanto, una exploración del algoritmo VAD basado en aprendizaje automático podría aumentar la robustez del sistema en presencia de ruido al momento de clasificar el sonido percibido por el micrófono.

Por otra parte, se propone estudiar otros enfoques más complejos como reconocimiento de sentencias haciendo uso de Modelos Ocultos de Márkov y Deep Learning, con esto se podrían obtener órdenes más complejas y con mayor vocabulario. También con esto se intentaría recrear sistemas más complejos con una cantidad de dispositivos aun mayor, lo cual representaría un control más complejo y a la vez más completo de la automatización de una casa.

Por otra parte, se propone explorar la programación del microcontrolador ESP8266 o similares para realizar no solo tareas de conmutación, sino también de atenuación de luces, control de otros dispositivos electrónicos tales como sistemas de riego, televisiones, sistemas de audio, control de posición de la cama donde se encuentra el usuario y todas las órdenes que a estos se le puedan emitir tal como subir o bajar el volumen, cambiar de canal o estación, etcétera. Con esto se propone crear una familia de dispositivos que puedan ser fácilmente integrados en el sistema actual con pocas

modificaciones al sistema actual y que permitan una diversidad más grande en las tareas posibles a realizar.

Bibliografía

Abd Wahab, M. (2016). IoT-based Home Automation System for People with Disabilities. *International Conference on Reliability, Infocom Technologies and Optimization*, (pág. 51). Noida.

Alsteris, L. D., & Paliwal, K. K. (2007). Short-time phase spectrum in speech processing: A review and some experimental results. *Digital Signal Processing* , 578-616.

Anusuya, M., & Katti, S. (2009). Speech Recognition by Machine: A Review. *International Journal of Computer Science and Information Security,* , 181-205.

Baraka, K., Ghobril, M., Malek, S., Kanj, R., & Kayssi, A. (2013). Low cost Arduino/Android-based Energy-Efficient Home Automation System with Smart Task Scheduling. *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, (págs. 296-301). Madrid.

Berger, A. (2002). *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques.* CMP Books.

Bray, J., & Sturman, C. F. (2002). *Bluetooth: connect without cables.* New Jersey: Prentice Hall.

Brown, K. (2005). *Encyclopedia of Language and Linguistics.* Elsevier.

Calaza, G. T. (2016). *Raspberry Pi2 para electrónicos.* Ciudad de México: Alfaomega.

Camargo, E., Coronel, C., & Calderón, M. (2015). SMART HOME CONTROL BY VOICE USING NEURAL NETWORKS. *Revista Colombiana de Tecnologias de Avanzada*, 17-20.

Chen, S.-C., Wu, C.-M., Chen, Y.-J., Chin, J.-T., & Chen, Y.-Y. (2017). Smart Home Control for the People with Severe Disabilities. *Proceedings of the 2017 IEEE International Conference on Applied System Innovation*, 503-506.

Cohen , M., Giangola, J., & Balogh, J. (2004). *Voice User Interface Design.* Addison-Wesley.

Darabkh, K. A., Khalifeh, A. F., Bathech, B. A., & Sabah, S. W. (2013). Efficient DTW-Based Speech Recognition System for Isolated Words of Arabic Language. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 586-593.

Davis, S. B., & Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences . *IEEE Transactions on acoustics, speech, and signal processing*, 357-366.

De Luna Ortega, A., Martínez Romo, J. C., & Mora González, M. (2006). Reconocimiento de Voz con Redes Neuronales, DTW y Modelos Ocultos de Markov. *Conciencia Tecnológica*, 1-6.

Dhingra, S., Nijhawan, G., & Pandit, P. (2013). Isolated Speech Recognition Using MFCC and DTW. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 4086-4092.

Domingo, M. (2012). An overview of the Internet of Things for people with disabilities. *Journal of Network and Computer Applications*, 584-589.

Edwards, C. (2013). Not-So-Humble Raspberry Pi gets Big Ideas. *Engineering & Technology*, 31-33.

European Telecommunications Standards Institute. (27 de Marzo de 2018). *ETSI.* Obtenido de Work Programme Detailed Report: http://www.etsi.org/deliver/etsi_es/201100_201199/201108/01.01.03_60/es_201108v010103p.pdf

Eyben, F., Weninger, F., Squartini, S., & Schuller, B. (2013). Real-life voice activity detection with LSTM recurrent neural networks and an application to hollywood movies. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (págs. 483-487). IEEE.

Flickenger, R. (2002). *Building Wireless Community Networks.* California: O´Reilly & Associates, Inc.

Ganchev, T., Fakotakis, N., & Kokkinakis, G. (2005). Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task. *Proceedings of the SPECOM-2005* (págs. 191-194). Patras: University of Patras.

Gast, M. S. (2002). *802.11 Wireless Networks.* California: O'Reilly & Associates, Inc.

Gelfand, S. (2016). *Essentials of Audiology.* New York: Thieme.

Ghazal, B., & Al-Khatib, K. (2015). Smart Home Automation System for Elderly, and Handicapped. *International Journal of Smart Home*, 203-210.

Hillar, G. C. (2017). *MQTT Essentials - A lightweight IoT protocol.* Birmingham: Packt Publishing.

Holmes, J., & Holmes, W. (2001). *Speech Synthesis and Recognition.* Nueva York: Taylor & Francis.

Huang, X., Baker, J., & Reddy, R. (2014). A Historical Perspective of Speech Recognition. *Communications of the ACM*, 95-103.

Hussein, A., Mehdi, A., Mirna, A., & Fahs, W. (2014). Smart Home Design for Disabled People based on Neural Networks. *The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks*, (págs. 117-126). Halifax.

Instituto Nacional de Estadística y Geografía. (18 de 10 de 2017). *INEGI.* Obtenido de Página Web del INEGI: http://internet.contenidos.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/nueva_estruc/702825090203.pdf

Instituto Nacional de las Personas Adultas Mayores. (22 de 9 de 2016). *INAPAM*. Obtenido de Página Web del INAOAM: https://www.gob.mx/inapam/galerias/estadisticas-sobre-adultos-mayores-en-mexico

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 67-72.

Jang, R. (8 de Agosto de 2017). *http://mirlab.org/jang/books/audioSignalProcessing/speechFeatureMfcc.asp?title=12-2%20MFCC*. Obtenido de Audio Signal Processing and Recognition (音訊處理與辨識): http://mirlab.org/jang/books/audioSignalProcessing/speechFeatureMfcc.asp?title=12-2%20MFCC

Kamarudin, N., Al-Haddad, S., Khmag, A., & bin Hassan, A. (2016). Analysis on Mel Frequency Cepstral Coefficients and Linear Predictive Cepstral Coefficients as Feature Extraction on Automatic Accents Identification. *International Journal of Applied Engineering Research*, 7301-7307.

Kishore , K., & P., P. (2016). Comparative Analysis of MFCC, LFCC, RASTA-PLP. *International Journal of Scientific Engineering and Research*, 4-7.

Kolokolov, A. S. (2003). Preprocessing and Segmentation of the Speech Signal in the Frequency Domain for Speech Recognition. *Automation and Remote Control*, 152-162.

Kyas, O. (2013). *How to Smart Home.* Wyk: Key Concept Press.

Lawson, A., Vabishchevich, P., Huggins, M., Ardis, P., Battles, B., & Stauffer, A. (2011). Survey and evaluation of acoustic features for speaker recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing* (págs. 5444-5447). Praga: IEEE.

Lippman, R. (1997). Speech recognition by machines and humans. *Speech Communication* , 1-15.

Loizou, P. C. (2013). *Speech Enhacement: Theory and Practice.* Boca Raton: CRC press.

Loweimi, E., Ahadi, S. M., Drugman, T., & Loveymi, S. (2013). On the Importance of Pre-emphasis and Window Shape in Phase-Based Speech Recognition. En T. Drugman, & T. Dutoit, *Advances in Non Linear Speech Processing* (págs. 160-167). Belgica: Springer.

Lv, X., & Zhang, M. (2008). Robot Control Based on Voice Command. *International Conference on Automation and Logistics*, (págs. 2490-2494). Qingdao.

Mahkonen, K. (8 de August de 2017). *SGN-4010 Puheenkäsittelyn menetelmät, 2 op.* Obtenido de Tampereen teknillinen yliopisto: www.cs.tut.fi/kurssit/SGN-4010/ikkunointi_en.pdf

Mittal, Y., Toshniwal, P., Sharma, S., Singhal, D., Gupta, R., & Mittal, V. (2015). A Voice-Controlled Multi-Functional Smart Home Automation System. *IEEE INDICON 2015*, (págs. 2-6). New Delhi .

Mohan, B. J., & Ramesh, N. B. (2014). Speech Recognition using MFCC and DTW. *2014 International Conference on Advances in Electrical Engineering* (págs. 1-4). Vellore: ICAEE.

Müller, M. (2007). *Information Retrieval for Music and Motion.* Berlin: Springer.

Mustafa, M. K., Allen, T., & Evett, L. (2014). A Review of Voice Activity Detection Techniques for On-Device Isolated Digit Recognition on Mobile Devices. En M. Bramer, & M. Petridis, *Research and Development in Intelligent Systems XXXI* (págs. 317-329). Suiza: Springer.

Nishimori, M., Saitoh, T., & Konishi, R. (2007). Voice Controlled Intelligent Wheelchair. *SICE Annual Conference 2007*, (págs. 336-340). Kagawa.

Oltenau, A.-C., Oprina, G.-D., Tapus, N., & Zeisberg, S. (2013). Enabling mobile devices for home automation using ZigBee. *2013 19th International Conference on Control Systems and Computer Science*, (págs. 189-195). Bucarest.

Paliwal, K. K. (1984). Effect of preemphasis on vowel recognition performance. *Speech Communication*, 101-106.

Paliwal, K. K., Agarwal, A., & Sinha, S. S. (1982). A Modification Over Sakoe And Chiba's Dynamic Time Warping Algorithm For Isolated Word Recognition. *Signal Processing*, 329-333.

Prabhu, B., & Pradeep, M. (2016). Implementation of Voice Recognition Wireless Home Automation System with ZigBee. *International Journal for Research on Extended Education*.

Rabiner, L., & Biing-Hwang, J. (1993). *Fundamentals of Speech Recognition.* Prentice Hall.

Ramirez, J., Gorriz, J. M., & Segura, J. C. (2007). Voice Activity Detection. Fundamentals and Speech Recognition System Robustness. En M. Grimm, & K. Kristian , *Robust Speech Recognition and Understanding* (págs. 1-22). InTech.

Ramlee, R. A., Leong, M. H., Singh, R. S., Ismail, M. M., Othman, M. A., Sulaiman, H. A., . . . Said, M. M. (2013). Bluetooth Remote Home Automation System Using Android Application. *The International Journal of Engineering And Science*, 149-153.

Raspberry Pi Foundation. (25 de Octubre de 2017). *What is a Raspberry Pi*. Obtenido de Raspberry Pi: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

Rojas-Rodríguez, R., Cortés-Aburto, O., Aceves-Pérez, R., Vela-Valdés, L., Arroyo Díaz, S., & García-Meneses, C. (2015). Home Automation Prototype Controlled. *International Journal of Combinatorial Optimization Problems and Informatics*, 54-63.

Saini, P., & Kaur, P. (2013). Automatic Speech Recognition: A Review. *International Journal of Engineering Trends and Technology*, 132-136.

Sakoe, H., & Chiba, S. (1978). Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 43-49.

Shaikh, H., Mesquita, L., & Das Chagas, S. (2017). Recognition of Isolated Spoken Words and Numeric using MFCC. *International Journal of Engineering Science and Computing*, 10539-10543.

Shin, J. W., Chang, J.-H., & Kim, N. S. (2010). Voice activity detection based on statistical models. *Computer Speech and Language*, 515-530.

Shuang-Hua, Y. (2014). *Wireless Sensor Networks*. Londres: Springer.

Slaney, M. (27 de Marzo de 2018). *Purdue University*. Obtenido de College of Engineering Purdue University: https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf

Taabish, G., Anand, S., & Sandeep, S. (2014). Comparative Analysis of LPCC, MFCC and BFCC for the Recognition of Hindi Words using Artificial Neural Networks. *International Journal of Computer Applications* , 22-27.

Verteletskaya, E., & Sakhnov, K. (2010). Voice Activity Detection for Speech Enhancement Applications. *Acta Polytechnica*, 100-105.

White, G. M., & Neely, R. B. (1976). Speech recognition experiments with linear predication, bandpass filtering, and dynamic programming. *IEEE Transactions on acoustics, speech, and signal processing*, 183-188.

Wortmann, F., & Flüchter, K. (2015). Internet of Things. *Business & Information Systems Engineering*.

Xuedong, H. (1993). On Speaker-Independent, Speaker-Dependent, and Speaker-Adaptive Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 150-157.

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research*, 724-735.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., . . . Woodland, P. (8 de August de 2017). *The HTK Book*. Obtenido de HTK 3: http://htk.eng.cam.ac.uk/docs/docs.shtml

Zhang, X.-L., & Wu, J. (2013). Deep Belief Networks Based Voice Activity Detection. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, 697-710.

Zheng, F., Zhang, G., & Song, Z. (2001). Comparison of different implementations of MFCC. *Journal Computer Science & Technology*, 582-589.

Apéndice A. Pasos para la Instalación del Broker MQTT

Primero, se importan las llaves que permitan que el software sea descargado e instalado para la versión de Raspbian instalada, esto se realiza introduciendo los siguientes comandos en la terminal de Raspbian:

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
sudo nano /etc/apt/sources.list.d/mosquitto.list
```

Una vez hecho lo anterior, se abrirá una pantalla con el software nano en el cual se muestran las fuentes de instalación (véase la Figura A.1).

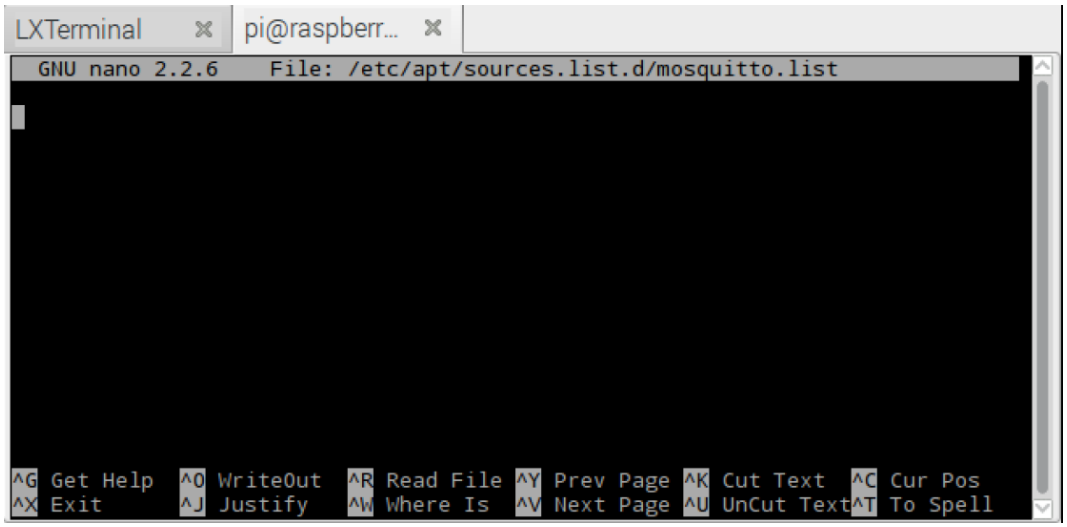


Figura A. 1. Archivo de configuración de fuentes.

Al archivo `mosquitto.list` se agrega la siguiente línea:

```
deb http://repo.mosquitto.org/debian jessie main
```

Después se actualizan las fuentes de instalación para poder tener acceso al nuevo software que puede ser instalado usando el siguiente comando:

```
sudo apt-get update
```

Finalmente, se procede a instalar el broker con el siguiente comando:

```
sudo apt-get install mosquitto mosquitto-clients
```

Como la tarjeta Raspberry Pi no realiza en este caso otra función que la de *broker*, no se necesita modificar la configuración por defecto de Mosquitto, cabe mencionar que el puerto configurado por defecto de la aplicación es el 1883. Este número de puerto se utilizará para las configuraciones de los dispositivos y de la aplicación móvil.

Otro parámetro necesario para que la configuración del *broker* quede finalizada es una dirección IP estática, a la cual se conectarán los dispositivos y la aplicación SAD. Para esto se abre una terminal y se introduce el siguiente comando.

```
ifconfig
```

Esto mostrará las interfaces de red disponibles como muestra la Figura A.2.

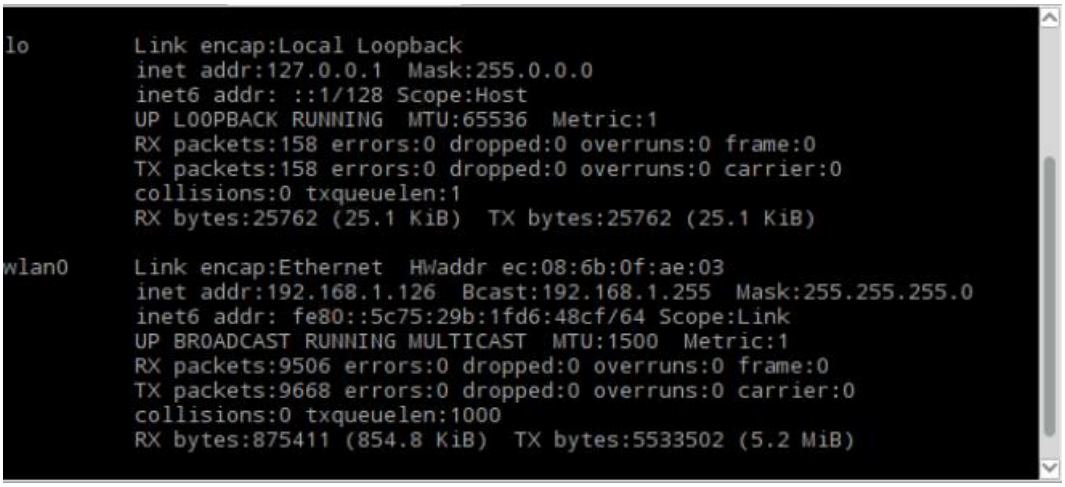


Figura A. 2. Interfaces disponibles mediante ifconfig.

En este caso se seleccionó el dispositivo wlan0 que corresponde a la tarjeta inalámbrica que utiliza la Raspberry Pi, a continuación, se configuran los parámetros de IP y puerta de enlace mediante los siguientes comandos.

```
sudo ifconfig wlan0 192.168.1.126 netmask 255.255.255.0
sudo route add default gw 192.168.1.254 wlan0
```

Se revisan los resultados de esta configuración mediante el uso del siguiente comando (véase Figura A.3).

```
route -n
```

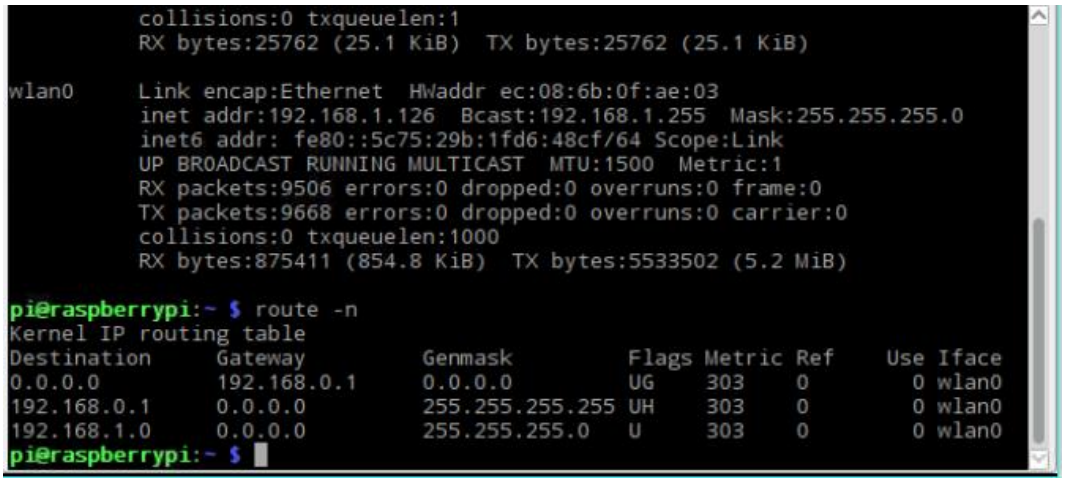


Figura A. 3. Resultado de la configuración.

Se obtiene la configuración final con la dirección IP 192.168.1.126, la cual es la dirección IP de conexión que usa el broker en el puerto TCP número 1883.

Una vez instalado Eclipse Mosquitto, se programa un *script* llamado mqstarto1.sh para ejecutarse al inicio de sesión de la plataforma, la tarea principal de este *script* es ejecutar una terminal sobre la cual se inicia el servicio de Mosquitto.

```
#/bin/bash
echo "INCIO MOSQUITTO"
lxterminal -e mosquitto -v
$SHELL
```

El script se agregó al inicio de cada sesión de la Raspberry PI con el siguiente comando:

```
nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

Una vez abierto el archivo `autostart`, se agrega la línea (véase Figura A.4):

```
./mqstarto1.sh .
```

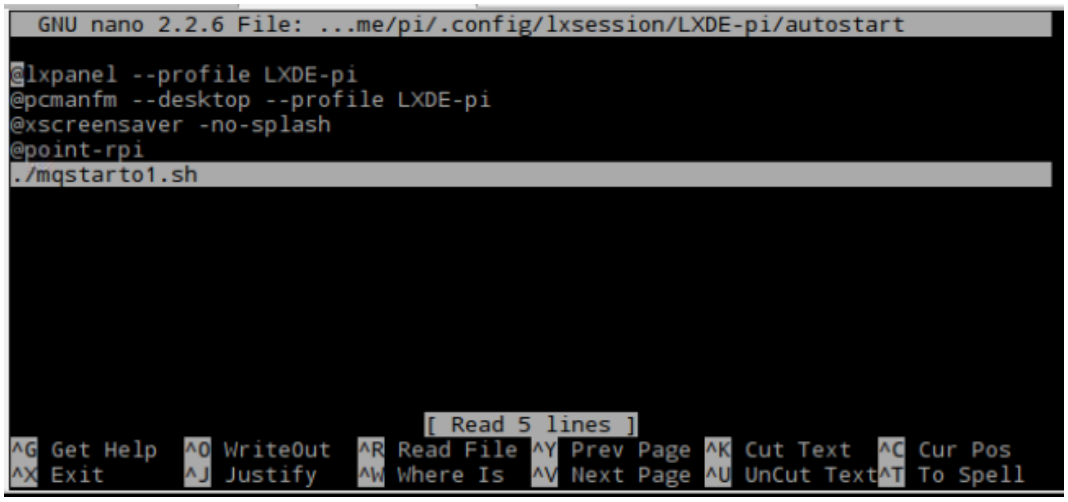


Figura A. 4. Archivo de configuración de autoarranque.

Finalmente, se procede a reiniciar la plataforma con el comando `reboot` y con ello queda completada la configuración del broker.

Apéndice B. Configuración de los Actuadores

Como se mencionó en la sección anterior, el dispositivo Sonoff Basic permite la actualización a otro firmware que no es propietario. Para esto, se necesita realizar un par de modificaciones físicas que permitan esta actualización del dispositivo.

Para permitir su programación se necesitan soldar conexiones en las terminales internas del dispositivo en 3V3, RX, TX, GND señaladas en la Figura B.1. Estas terminales corresponden a las terminales de una tarjeta serial utilizada para programar. Se procede a conectar las tarjetas en los pines correspondientes mostrados en la Figura B.2 y reprogramar los actuadores utilizando el software `esptool`.

Figura B. 1. Interior de Sonoff Basic, con conexiones.

La Figura B.2 muestra el diagrama de conexión entre el Sonoff Basic y la tarjeta serial.

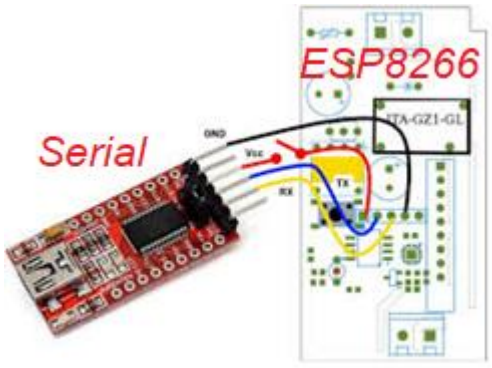


Figura B. 2. Diagrama de conexiones para reprogramación, la tarjeta roja es la tarjeta serial y la tarjeta blanca el dispositivo Sonoff Basic.

Para comenzar la reprogramación se requiere que se presione el botón GPIO0 cuando la tarjeta sea conectada a una fuente de alimentación, debido a que se necesita que el puerto GPIO0 esté conectado a GND para entrar en el modo de programación del

microcontrolador. Una vez terminado este proceso, se abre una terminal de Windows y se introduce la siguiente línea incluyendo el archivo de firmware a instalar (véase Figura B.3):

```
esptool.exe -vv -cd nodemcu -cb 115200 -cp COM3 -ca 0x00000 -cf
```

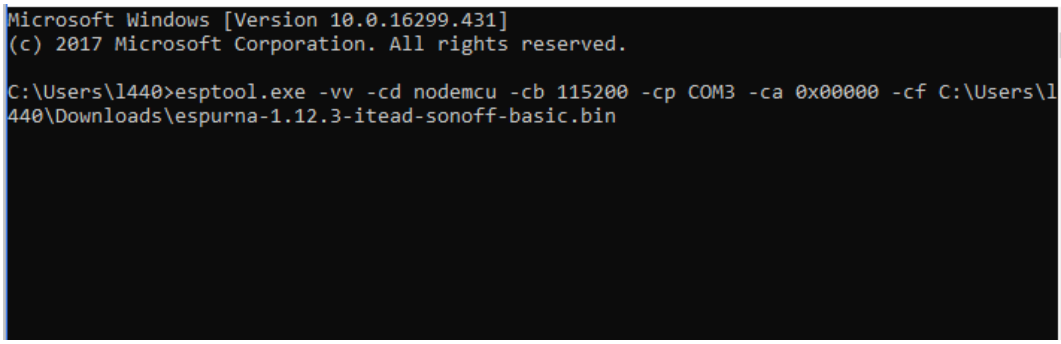


Figura B. 3. Programación de firmware.

Para iniciar la programación del nuevo firmware, se oprime la tecla Enter, en este caso la versión seleccionada de ESPurna es el apartado 1.12.3.

Una vez programado el firmware, se procede a configurar el actuador Sonoff Basic para hacerlo capaz de comunicarse con el *broker*. Para comprobar la programación exitosa debe de tener acceso al formulario de inicio de sesión del dispositivo, este dispositivo debe ser visible como red Wi-Fi y debe de permitir conectarse a él. A partir de esto se comienza su configuración. En la Figura B.4 se muestra la pantalla de configuración del dispositivo una vez concluida la programación exitosa.

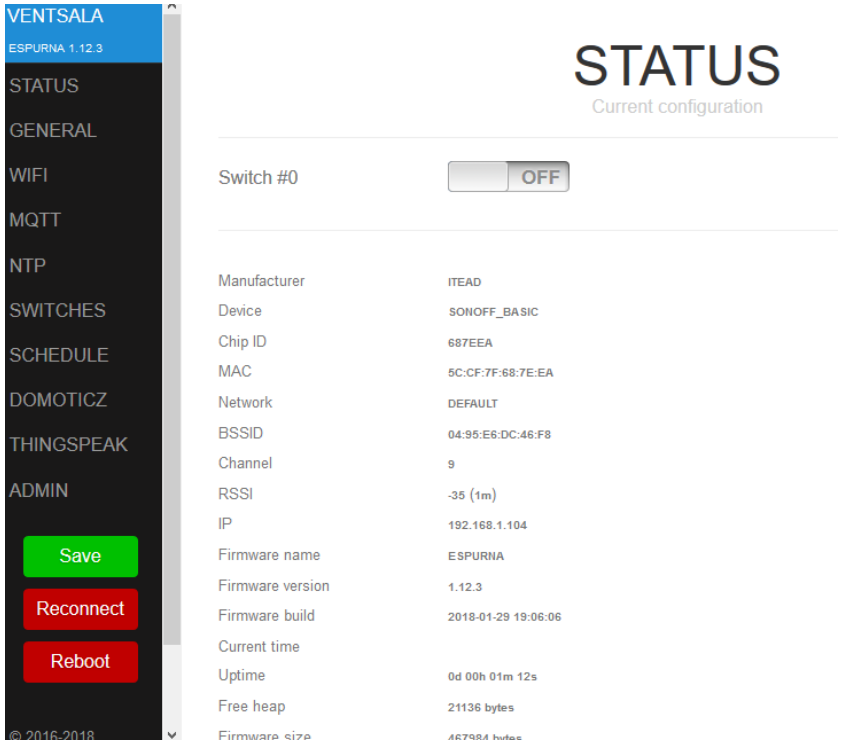


Figura B. 4. Página de configuración de dispositivo.

A continuación, se procede a editar la configuración inalámbrica (véase Figura B.5).

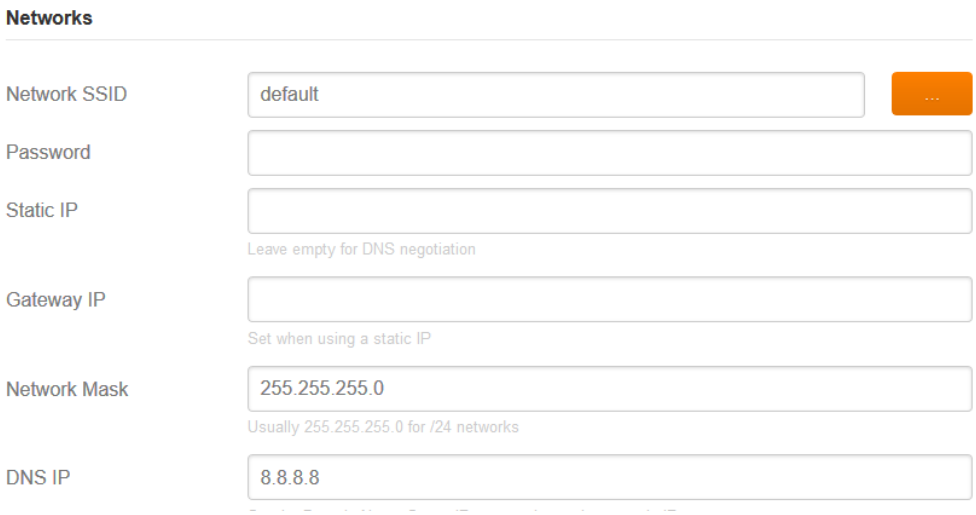


Figura B. 5. Configuración inalámbrica del dispositivo.

A continuación, se configura el servicio de MQTT para poder interactuar con el *broker*, para esto se usan los datos configurados y se realizan las modificaciones necesarias tal como se presenta en la Figura B.6.

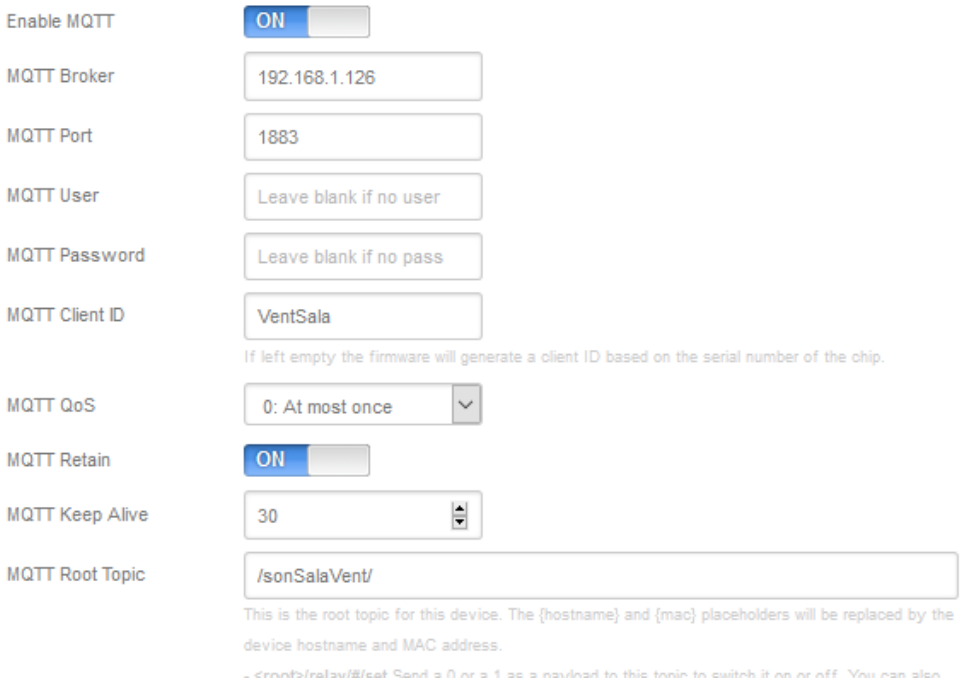


Figura B. 6. Configuración de protocolo de comunicaciones MQTT.

Con esto se asigna un nombre de cliente basado en un esquema que contiene el lugar donde se encuentra el actuador y el tipo de aparato a controlar, los parámetros de configuración, el nivel de QoS y el t*opic*, este último es importante para el reconocimiento de órdenes. Finalmente se guardan los cambios y se procede a probar las configuraciones; cabe señalar que este proceso se debe llevar a cabo para cada uno de los dispositivos que se deseen agregar al esquema.

Apéndice C. Algoritmos Simplificados

En esta sección se presenta una simplificación de los algoritmos presentados con la intención de ayudar a entender su comportamiento.

Algoritmo MFCC

Entrada:

Señal de audio procesada por el algoritmo VAD = S

Frecuencia de muestreo = fs

Tiempo de ventana = t

Coeficientes de preénfasis = Alpha

Frecuencia inicial de filtros = fi

Frecuencia final de filtros = ff

Número de coeficientes = numcoeff

Salida:

Coeficientes MFCC = coeffin

1. A S se le aplica el preénfasis con los coeficientes Alpha.
2. Al resultado de lo anterior se le obtiene su espectrograma con los parámetros fs y t.
3. Se crea un Banco de filtros BA con los parámetros fs, fi, ff y t.
4. Se declara una matriz llamada filterOut que tiene una dimensión del número de filas del espectrograma por el número de filtros.
5. Se declara un vector llamado sums igual al número de filas del espectrograma.
6. Para i=0 hasta el número de filas del espectrograma.
   6.1. Se le aplica el filtrado con BA a la fila i del espectrograma y se almacena en filterOut de i.
   6.2. A este resultado del filtrado se le realiza la suma de todos los elementos y se guarda en sums de i.
   6.3. A cada uno de los resultados obtenidos de filterOut de i se le aplica su logaritmo natural y se reemplaza el valor antiguo por el valor actual.
7. Se declara una matriz llamada coeff que tiene una dimensión del número de filas de filterOut por el valor de numcoeff.
8. Para i=0 hasta el número de filas de filterOut
   8.1. Se aplica la DCT y se recortan los elementos para filterOut de i y se almacenan en coeff de i.
9. Para i=0 hasta el número de filas de coeff
   9.1. Se reemplaza el primer coeficiente en coeff con el logaritmo del elemento sums de i.

10. Se declara una matriz deltas en donde se almacena el resultado de la aplicación del cálculo de deltas usando como entrada la matriz coeff.
11. Se declara una matriz deltadeltas en donde se almacena el resultado de la aplicación del cálculo de deltas usando como entrada la matriz deltas.
12. Se concatenan todas las matrices coeff, deltas y deltadeltas para formar coeffin.
13. Se regresa la matriz coeffin.

Algoritmo DTW

Entradas:

Matriz de características 1 = A

Matriz de características 2 = B

Tamaño de la banda = wi

Salida:

Distancia calculada=d

1. Se verifica que A y B tengan el mismo número de filas.
   1.1. Si esto falla d=NaN
2. Se verifica que el tamaño de banda sea válido.
   2.1. Si esto falla la wi es igual a la diferencia del número de columnas entre A y B.
3. Se declara una matriz D la cual tiene una dimensión del número de columnas de A por el número de columnas de B.
4. Se inicializa la matriz D con infinito para cada uno de los elementos.
5. Se define el primer elemento de la primera columna en D para que este sea 0;
6. Repetir i =0 hasta que D este en la última fila.
   6.1. Repetir j=0 mientras este se encuentre en un rango de wi valido.
      6.1.1. Se calcula la distancia entre la fila i de A y la fila j de B.
      6.1.2. Se actualiza D de i de j con base a la ecuación de recursión que le sea aplicada.
7. Se asigna a d el valor de D en la última fila y en la última columna.

Algoritmo VAD

Entradas:

Señal para procesar = S

Límite de cruces por 0 = thX

Límite de Energía =thE

Límite de correlación en Lag 1 =thCorr

Tamaño de ventana = windZ

Salida:

Señal procesada = SV

1. Se declara un arreglo variable SV.
2. Se divide el tamaño de la señal S entre windZ para obtener el número de segmentos no superpuestos nf.
3. Repetir de i=0 hasta nf
   3.1. Se inicializa el número de banderas activas nm=0.
   3.2. Se extraen windZ elementos de señal y se almacenan en tmp.
   3.3. Se aplica la FFT y se almacena en un arreglo llamado En.
   3.4. Se aplica el absoluto para el arreglo En y se guarda ahí mismo
   3.5. Se declara sumE =0.
   3.6. Se realiza la sumatoria de En y se guarda en sumE.
   3.7. Si SumE es mayor a thE
      3.7.1. nm = nm+1;
   3.8. A tmp se le aplica la autocorrelación en Lag 1 y se guarda en autoX.
   3.9. Si autoCorr es mayor a thCorr
      3.9.1. nm=nm+1;
   3.10. A tmp se le aplica el conteo de cruces por cero y se guarda en numX.
   3.11. Si numX es menor a thX
      3.11.1. nm=nm+1;
   3.12. Si nm es mayor a 2
      3.12.1. tmp se concatena al arreglo variable SV.
4. Se regresa SV