Neiva, 4 de Febrero de 2019

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Neiva, Huila

El (Los) suscrito(s):

LUIS FELIPE GONZÁLEZ CORTES , con C.C. No. 1.081.182.821

ANA YULIETH PERDOMO PERDOMO , con C.C. No. 1.075.272.950

_____, con C.C. No. _____,

_____, con C.C. No. _____,

autor(es) de la tesis y/o trabajo de grado o _____

titulado MÓDULO DE RECONOCIMIENTO DE PALABRAS BASADO EN LPC PARA EL CONTROL DE DESPLAZAMIENTO DE DISPOSITIVOS CON MICROMOTORES presentado y aprobado en el año 2019 como requisito para optar al título de

INGENIERO ELECTRÓNICO ;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Luis Felipe González Cortés

Firma: Luis F. Gonzalez _____

EL AUTOR/ESTUDIANTE:

Ana Yulieth Perdomo Perdomo

Firma: Ana Yulieth Perdomo P. _____

EL AUTOR/ESTUDIANTE:

Firma: _____

EL AUTOR/ESTUDIANTE:

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: Módulo de reconocimiento de palabras basado en LPC para el control de desplazamiento de dispositivos con micromotores.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
González Cortés	Luis Felipe
Perdomo Perdomo	Ana Yulieth

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Bravo Obando	Martin Diomedes

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
Molina Mosquera	Johan Julian
Mosquera Cerquera	Vladimir

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2019 **NÚMERO DE PÁGINAS:** 81

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas **X** Fotografías **X** Grabaciones en discos___ Ilustraciones en general **X** Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos__ Retratos___ Sin ilustraciones___ Tablas o Cuadros **X**

SOFTWARE requerido y/o especializado para la lectura del documento: Lector PDF.

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. LPC	LPC	6.	
2. Extracción de características	Feature extraction	7.	
3. Patrones	Patterns	8.	
4. SVD	SVD	9.	
5. Distancia Euclidiana.	Euclidean distance	10.	

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

En la actualidad distintas aplicaciones tecnológicas han mostrado grandes avances enfocados al reconocimiento del habla aplicado al control de procesos utilizando métodos estadísticos o matemáticos para la identificación de los patrones de la voz. De esta manera, un patrón de voz de entrada es representada como una secuencia de vectores de características acústicas.

El siguiente libro describe el diseño e implementación de un sistema de reconocimiento de palabras con el propósito de controlar el movimiento de un modelo escala de silla de ruedas aplicando como método matemático LPC para la extracción de características de la señal de voz y la función de distancia euclidiana para la comparación de patrones. Para la implementación del módulo se usa la raspberry pi, la cual se implementa el algoritmo de reconocimiento de los comandos y la ejecución del control de los servomotores.

Vigilada mieducación
La versión vigente y controlada de este documento, solo podrá ser consultada a través del sitio web Institucional www.usco.edu.co, link Sistema Gestión de Calidad. La copia o impresión diferente a la publicada, será considerada como documento no controlado y su uso indebido no es de responsabilidad de la Universidad Surcolombiana.

ABSTRACT: (Máximo 250 palabras)

Currently, different technological applications have shown great advances focused on the recognition of speech applied to process control using statistical or mathematical methods for the identification of voice patterns. In this way, an input speech pattern is represented as a sequence of vectors of acoustic characteristics.

This document presents the design and implementation of a word recognition system with the purpose of controlling the movement of a wheelchair scale model, applying as a reference method, the linear prediction coefficients (LPC) for the extraction of the characteristics of the voice signal and in turn the use of the Euclidean distance function for the patterns comparison. For the implementation a module of raspberry pi was used, in which the algorithm of recognition of the commands and the execution of the control of the servomotors was implemented

APROBACIÓN DE LA TESIS

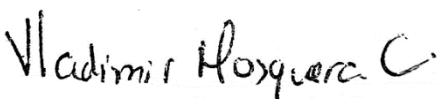
Nombre Presidente Jurado: Martín Diomedes Bravo Obando

Firma: 

Nombre Jurado: Johan Julián Molina Mosquera

Firma: 

Nombre Jurado: Vladimir Mosquera Cerquera

Firma: 

MÓDULO DE RECONOCIMIENTO DE PALABRAS BASADO EN LPC PARA EL CONTROL DE DESPLAZAMIENTO DE DISPOSITIVOS CON MICROMOTORES

LUIS FELIPE GONZÁLEZ CORTÉS
ANA YULIETH PERDOMO PERDOMO

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA
2018

MÓDULO DE RECONOCIMIENTO DE PALABRAS BASADO EN LPC PARA EL CONTROL DE DESPLAZAMIENTO DE DISPOSITIVOS CON MICROMOTORES

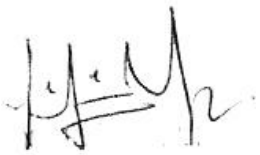
LUIS FELIPE GONZÁLEZ CORTÉS
ANA YULIETH PERDOMO PERDOMO

Proyecto de grado para optar al título de Ingeniero Electrónico

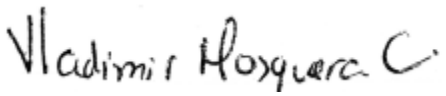
Director
Martín Diomedes Bravo Obando
Magíster en Telecomunicaciones

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA
2018

Nota de aceptación:
Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Surcolombiana para optar al título de Ingeniero Electrónico



Jurado: Johan Julián Molina M.



Jurado: Vladimir Mosquera

Neiva, 2019

AGRADECIMIENTOS

A Dios, por permitir la culminación de este logro tan importante en nuestra formación profesional, a nuestros padres por el apoyo incondicional durante todo el proceso, al ingeniero Martín Diomedes Bravo Obando por compartirnos sus conocimientos para el desarrollo del proyecto y a todos los demás ingenieros que hicieron parte del proceso culminado.

RESUMEN

En la actualidad distintas aplicaciones tecnológicas han mostrado grandes avances enfocados al reconocimiento del habla aplicado al control de procesos utilizando métodos estadísticos o matemáticos para la identificación de los patrones de la voz. De esta manera, un patrón de voz de entrada es representada como una secuencia de vectores de características acústicas.

El siguiente libro describe el diseño e implementación de un sistema de reconocimiento de palabras con el propósito de controlar el movimiento de un modelo escala de silla de ruedas aplicando como método matemático LPC para la extracción de características de la señal de voz y la función de distancia euclidiana para la comparación de patrones. Para la implementación del módulo se usa la raspberry pi, la cual se implementa el algoritmo de reconocimiento de los comandos y la ejecución del control de los servomotores.

PALABRAS CLAVES: LPC. Extracción de características. Patrones. SVD. Distancia Euclidiana.

ABSTRACT

Currently, different technological applications have shown great advances focused on the recognition of speech applied to process control using statistical or mathematical methods for the identification of voice patterns. In this way, an input speech pattern is represented as a sequence of vectors of acoustic characteristics.

This document presents the design and implementation of a word recognition system with the purpose of controlling the movement of a wheelchair scale model, applying as a reference method, the linear prediction coefficients (LPC) for the extraction of the characteristics of the voice signal and in turn the use of the Euclidean distance function for the patterns comparison. For the implementation a module of raspberry pi was used, in which the algorithm of recognition of the commands and the execution of the control of the servomotors was implemented.

Keywords: LPC. Feature extraction. Patterns. SVD. Euclidean distance.

CONTENIDO

Lista de figuras

Pág.

Lista de Tablas

Lista de gráficos

INTRODUCCIÓN

El habla es un acto individual por medio del cual un humano puede transmitir información y de esta manera poder comunicarse actualmente el procesamiento de señales de voz permite la interacción hombre a máquina creando así un sistema de reconocimiento, gracias a este avance existe una gran variedad de aplicaciones que se pueden modificar de acuerdo al interés del usuario.

Las nuevas tecnologías para el reconocimiento de voz están revolucionando la forma en que el usuario recibe y procesa la información automatizando sus diferentes actividades cotidianas, involucrando dispositivos como las computadoras, teléfonos, electrodomésticos, entre otros, esto conlleva a una mejora en la calidad de vida si se piensa en las personas con alguna discapacidad. Cuando se habla de un control con voz lo primero que se considera es el reconocimiento, es decir, que el sistema entienda las palabras mas no su significado y con base en este ejecuta una acción.

Para la realización del diseño de un módulo de reconocimiento de palabras es importante considerar cómo está constituido el sistema vocal humano para así ejecutar la analogía con un sistema digital y de esta manera dar hincapié al desarrollo del algoritmo que permita el proceso de reconocimiento y ejecución de acuerdo al comando dado por el usuario. El modelo LPC (Codificación de predicción lineal) es considerado como uno de los más próximos al sistema vocal analógicamente hablando y debido a ello se establecerá en el siguiente documento como punto clave para el proyecto.

El control de micromotores se lleva a cabo en un prototipo de silla de ruedas, también puede ser aplicado a diferentes modelos con dominios eléctricos y mecánicos que permita la realización de un desplazamiento, de esta manera los comandos descritos por el usuario será derecha, izquierda, atrás, adelante y alto.

1. ANALISIS PREVIO

1.1 Planteamiento del problema

Desde la antigüedad el ser humano es un ente sociable que hace uso de la voz como principal forma de comunicación, mediante el paso de los años y la aparición de máquinas como base para la realización de diferentes tareas en la actualidad, el hombre busca facilitar la interacción hombre-máquina de una manera cada vez más eficiente. “El habla es el principal medio de comunicación entre los seres humanos. Por razones que van desde la curiosidad tecnológica sobre los mecanismos para la realización mecánica de las capacidades del habla humana hasta el deseo de automatizar las tareas simples que requieren las interacciones entre hombre-máquina, la investigación en el reconocimiento automático del habla ha atraído la atención durante cinco décadas”. [1]

Aun con años de investigación y con el desarrollo de la precisión del habla automática, el reconocimiento de voz sigue siendo uno de los desafíos de investigación más importantes y requiere atención en las siguientes cuestiones: definición de varios tipos y clases de habla, representación del habla, función extracción, técnicas, clasificadores de voz, base de datos y rendimiento evaluación. [2]

Aplicar un modelo eficiente para el reconocimiento de voz es una tarea compleja ya que se debe tener en cuenta muchos factores importantes para que su desempeño sea óptimo, como por ejemplo el entorno donde se implementa. En este proyecto se propone diseñar un módulo de reconocimiento de palabras en donde el propósito es el manejo de un prototipo de silla de ruedas empleando tecnología reciente para reafirmar el nuevo concepto en cuanto a movilidad personalizada.

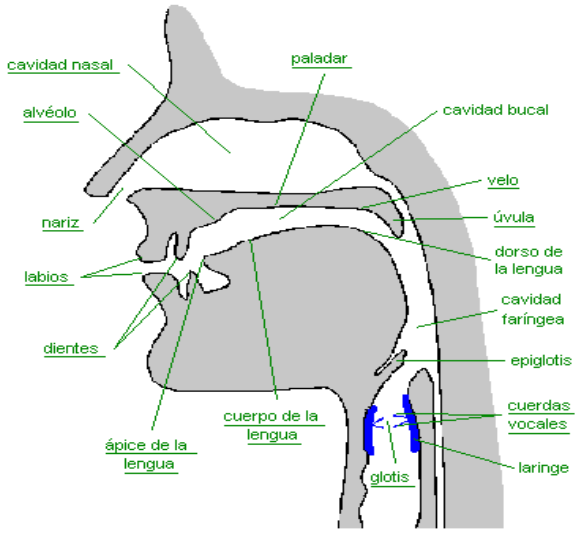
A la hora de diseñar e implementar un sistema de voz es necesario identificar los parámetros y las características tomando como base métodos matemáticos como LPC (Codificación Predictiva Lineal) para la extracción de características de la voz y a su vez para el reconocimiento se utiliza una medida de distancia (euclidiana) para la comparación de patrones caracterizados como referencia que están almacenados en memoria. Como resultado se obtiene un producto propio de la universidad Surcolombiana, un módulo básico de reconocimiento de palabras para el control de desplazamiento en dispositivos.

1.2 Señal de voz

1.2.1 Anatomía del aparato fonador

El conjunto de órganos anatómicos que intervienen en la producción de la voz se denomina aparato fonador y está formado por los pulmones como fuente de energía mediante el flujo de aire, la laringe que contiene las cuerdas vocales, la faringe, la cavidad oral, nasal y una serie de elementos articulatorios: los labios, los dientes, el alveolo, el paladar, el velo del paladar y la lengua como se muestra en la Figura 1. Fisiológicamente, la voz corresponde a la vibración de las cuerdas vocales, contenidas en la laringe, que a su vez es el órgano fonador por excelencia. Estos pliegues vocales, por medio de un proceso de aducción (cierre de los pliegues vocales al juntarse), ofrecen resistencia al flujo continuo de aire, obteniendo como resultado el sonido[3].

Figura 1. Aparato Fonador



Fuente: http://paginaspersonales.deusto.es/airibar/Fonetica/Apuntes/02.html

1.2.2 Características acústicas de los sonidos del habla

La onda sonora causada en la fonación es el resultado del paso del aire por la glotis en la emisión de una serie de exhalaciones al ritmo de la abertura y cierre de los pliegues vocálicos.

Para la producción de la onda sonora las moléculas de aire deben entrar en vibración, lo que se consigue por su paso a través de los pliegues vocálicos.

El espectro de la onda sonora presenta una amplitud descendente en los armónicos a medida que aumenta la frecuencia debido al fenómeno de la resonancia producida por el paso de la onda sonora por las cavidades superglóticas.

El formante es una zona de la escala de frecuencias en la que un sonido presenta una mayor concentración de energía. También puede definirse como cada una de las resonancias del conducto vocal[4]. Técnicamente los formantes son bandas de frecuencia donde se concentra la mayor parte de la energía sonora de un sonido. Estas resonancias o formantes son descritos según tres parámetros: el centro de frecuencia, ancho de banda y energía. Al modificar la forma del tracto vocal se modifican estos tres elementos en diferente medida y por lo tanto la función de transferencia y el sonido final cambiara. Los armónicos provenientes del sonido laríngeo serán reforzados o atenuados por estas resonancias formantes. De esta forma, los armónicos cercanos a los valores formanticos serán más amplificados que los armónicos que se encuentren más alejados de los formantes[5]. A continuación en la Tabla 1, se muestra algunos anchos de bandas entre las cuales se localizan las vocales.

Tabla 1. Formantes Vocálicos

Formantes Vocálicos	
Vocal	Región principal formantica
/u/	200 a 400 Hz
/o/	400 a 600 Hz
/a/	800 a 1200 Hz
/e/	400 a 600 y 2200 a 2600 Hz
/i/	200 a 400 y 3000 a 3500 Hz

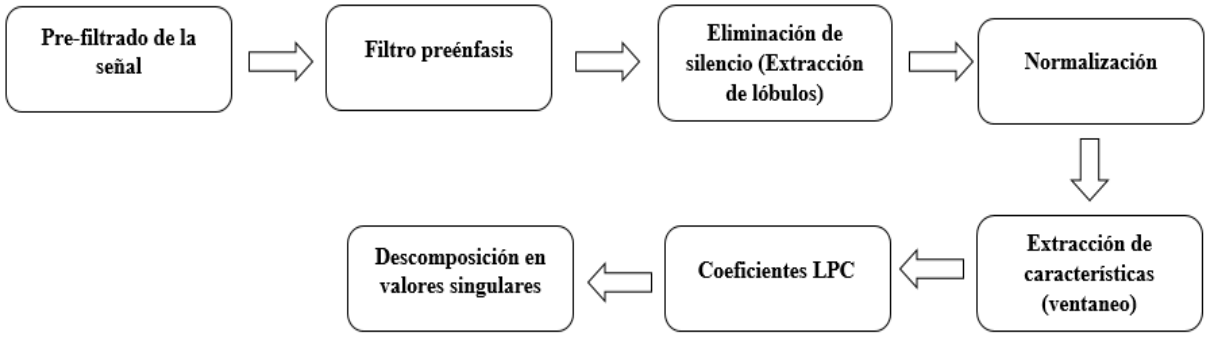
Fuente: https://es.wikipedia.org/wiki/Formante

2. REQUERIMIENTOS DEL SISTEMA

2.1 Procesamiento de la señal de voz

Los sistemas de reconocimiento de voz comprenden diferentes disciplinas y tienen en común la etapa inicial conocida como el procesamiento de señales, el cual convierte la señal de voz en alguna representación paramétrica para su posterior análisis, en la figura 2 se muestra la etapa de procesamiento de la señal de voz común entre la etapa de entrenamiento y reconocimiento usado en este trabajo.

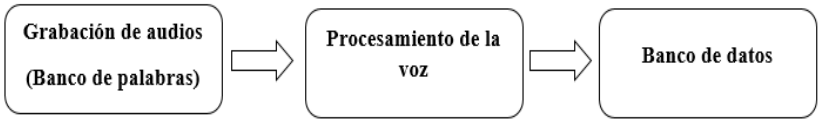
Figura 2. Procesamiento de la voz



Fuente: Autores

La etapa de entrenamiento realiza la extracción de patrones de cada uno de los comandos del banco de palabras y los guarda en un banco de datos para su posterior comparación como se muestra en la figura 4.

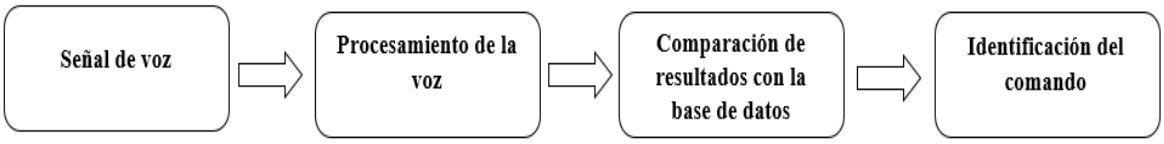
Figura 3. Etapa de entrenamiento



Fuente: Autores

La figura 4 muestra la etapa de reconocimiento la cual realiza la extracción de características del comando dicho por el usuario y lo compara con el banco de palabras por medio de la distancia euclidiana.

Figura 4. Etapa de reconocimiento



Fuente: Autores

2.1.1 Obtención del banco de palabras

El sistema de reconocimiento tiene cinco comandos para su funcionamiento (derecha, izquierda, atrás, alto y adelante), la señal de voz se muestrea a un rango de frecuencias entre 8 y 16 KHz. Para el desarrollo del algoritmo se realizaron grabaciones con un tiempo de 2 segundos a una frecuencia de 11025 Hz, se utilizó un formato WAV, y se tomaron muestras de hombres y mujeres de diferentes edades.

2.1.2 Pre-filtrado de la señal

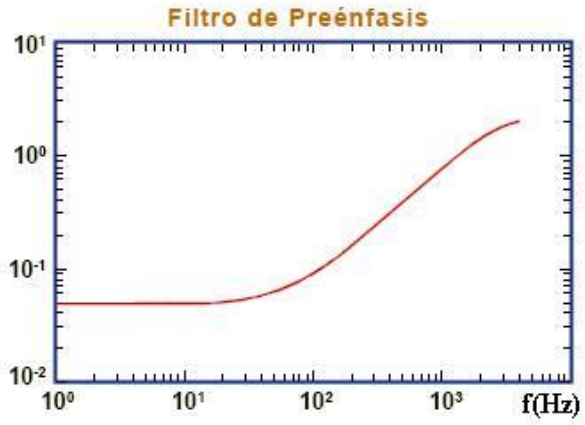
En sistemas de comunicación, los filtros se utilizan para sintonizar frecuencias específicas y eliminar otras.

El filtro butterworth es un tipo de diseño de filtro de procesamiento de señal y está diseñado para tener una respuesta de frecuencia lo más plana matemáticamente posible en la banda de paso[6]. De esta manera la realización de un pre-filtrado a las señales de voz utilizando este tipo de filtro pasa banda de orden 4th con frecuencias de corte 40 Hz y 1000 Hz es necesario con el fin de eliminar ciertas perturbaciones y soportar el proceso de reconocimiento.

2.1.3 Filtro preénfasis

Un filtro preénfasis es un método de procesamiento de señales diseñado para aumentar la magnitud de algunas frecuencias con respecto a las de otras con el fin de mejorar la relación señal a ruido, disminuyendo así los efectos como la atenuación y distorsión de la señal. En la Figura 5 se muestra el filtro en el dominio frecuencial.

Figura 5. Grafica del filtro preénfasis en el dominio frecuencial



Fuente:https://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_05_06/io3/public_html/procesamiento/reconocimiento/reconocimiento_04.html

El filtro preénfasis de primer orden aplicado está dado por la ecuación 1.

Ecuación 1. Filtro preénfasis

$$H(z) = (1 - az^{-1})$$

Donde a es igual 0.95 debido a que se necesita un filtro pasa alto, la aplicación de este filtro asegura al sistema espectralmente aplane la señal y no se pierda información[7].

2.1.4 Eliminación de silencio (Extracción de lóbulos)

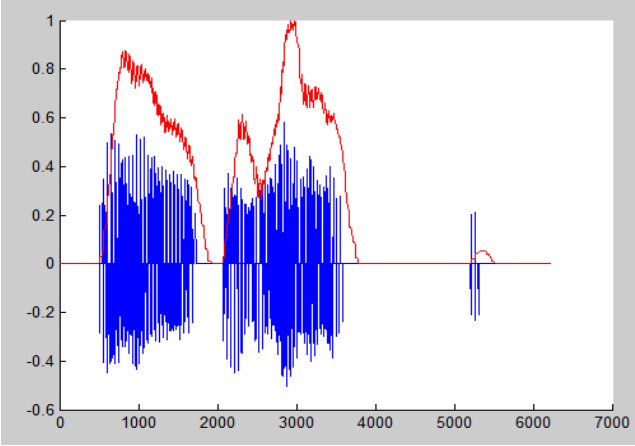
La palabra lóbulo en este trabajo hace referencia a un tramo de la señal de voz que identifica un fonema. Los segmentos de silencio de la señal digital son removidos tomando como base un valor umbral, extrayendo solo las partes con mayor energía para su posterior análisis. Para el cálculo de la energía se aplica en la ecuación 2.

Ecuación 2. Energía

$$E = \sum_{k=1}^{N} x[k]^2$$

donde $x[k]$ es la señal de entrada, k la posición de cada muestra, N es el número de muestras y E la salida[8].

Figura 6. Energía de la palabra "DERECHA"



Fuente: Autores

En la figura 6 se muestra en color rojo la energía de la palabra derecha, se puede observar cómo se diferencia un lóbulo del otro por medio de segmentos con valores en cero, esto ayuda a identificar el rango de los lóbulos, en este caso tiene tres lóbulos en total.

2.1.5 Normalización

La normalización es una operación estadística y se usa para escalar valores heterogéneos, en este caso aplicarlo facilita la definición de umbrales en diferentes algoritmos. Los valores de los datos se limitan desde -1 hasta 1. La ecuación que define este proceso está dada por la ecuación 3.

Ecuación 3. Normalización

$$Norma = x/\max(x)$$

donde x representa cada valor de la señal[9].

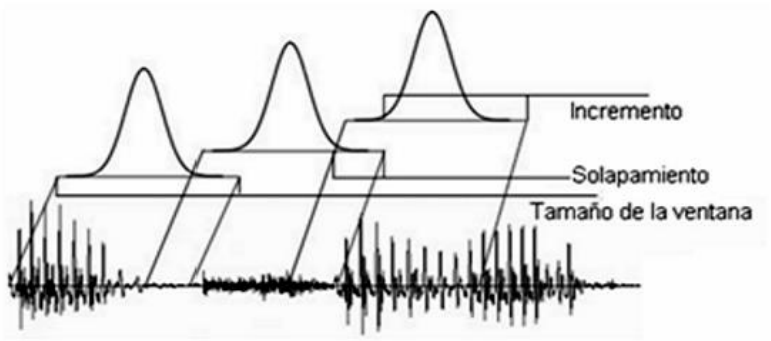
2.1.6 Ventaneo

En el sistema de reconocimiento de palabras se cuenta con grabaciones de diferentes personas sometidas a un pre-procesado adecuado, para así proceder a la extracción de

características de la señal de voz de acuerdo a ciertos parámetros.

Se entiende que la señal de voz contiene numerosas variaciones y debido a ello la extracción de características se debe realizar en intervalos entre 20 ms y 30 ms[10], para esto es oportuno aplicar una ventana a la señal de un tamaño conveniente de acuerdo a la frecuencia de muestreo en este caso se utiliza una frecuencia de muestreo de 11025 Hz a intervalos de 30 ms lo que equivale a 330 muestras por ventana con un incremento de 110 muestras aproximadamente.

Figura 7. Ventaneo de una señal



Fuente: http://orinoquia.unillanos.edu.co/index.php/orinoquia/article/view/272/804

La ventana utilizada en el proceso es el tipo Hamming y está dado por la ecuación 4.

Ecuación 4. Ventana Hamming

$$V(n) = \begin{cases} 0.54 - 0.46\, x \cos\left(\frac{2\pi n}{N}\right) & si\ 0 \leq n \leq N \\ 0 & En\ otro\ caso \end{cases}$$

donde n representa los valores de la señal y N el número de muestras[11].

2.1.7 Coeficientes LPC

Al emplear el ventaneo posteriormente se procede a aplicar las técnicas de extracción de patrones que son los coeficientes LPC (codificación por predicción lineal), sus parámetros se ajustan a las características del tracto vocal y representa la envolvente espectral de la señal de forma comprimida. Partiendo de la idea que se puede predecir la muestra presente a partir de una combinación lineal de las muestras pasadas, se genera una descripción espectral basada en

segmentos cortos de señal, considerando una señal s[n] a una respuesta de un filtro todo-polo de una excitación u[n]. La función de transferencia del filtro se describe en la ecuación 5.

Ecuación 5. Función de transferencia del filtro LPC

$$H(z) = \frac{Y(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{G}{A(z)}$$

donde G es el parámetro de ganancia, a_k los coeficientes del filtro y p el orden de dicho filtro.

La estimación de los coeficientes de predicción se obtiene minimizando el error de predicción[12], dado por la ecuación 6.

Ecuación 6. Error de predicción

$$e[z] = y[z](1 - \sum_{k=1}^{p} a_k z^{-k})$$

Para hallar los coeficientes LPC (a_k) se hace mediante diferentes métodos computacionales, en este caso el método utilizado es el Prony. El método de Prony extrae información valiosa de una señal muestreada uniformemente y construye una serie de exponenciales o sinusoidales complejas amortiguados. Esto permite la estimación de componentes de frecuencia, amplitud, fase y amortiguamiento de una señal. Suponiendo que las N muestras de datos complejos $x[1], \dots, x[N]$. la función investigada pueden aproximarse por M funciones exponenciales, el método de Prony está representado por la ecuación 7.

Ecuación 7. Función método de Prony

$$y[n] = \sum_{K=1}^{M} A_K e^{(a_k + jw_k)(n-1)Tp + j\psi_k}$$

donde Tp es el periodo de muestreo, A_K es la amplitud , a_k el factor de amortiguamiento, w_k la velocidad angular , ψ_k la fase inicial y n = 1,2,...,N.

La función en tiempo discreto esta expresada como se muestra en la ecuación 8.

Ecuación 8. Función método de Prony tiempo discreto

$$y[n] = \sum_{k=1}^{M} h_k z_k^{n-1}$$

donde

$$h_k = A_k e^{j\psi k}$$
$$z_k = e^{(a_k + jw_k)Tp}$$

La ecuación 8 puede ser expresada por una matriz como:

$$\begin{bmatrix} z_1^0 & z_2^0 & \cdots & z_M^0 \\ z_1^1 & z_2^1 & \cdots & z_M^1 \\ \vdots & \vdots & \cdots & \vdots \\ z_1^{M-1} & z_2^{M-1} & \cdots & z_M^{M-1} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_M \end{bmatrix} = \begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[M] \end{bmatrix}$$

La ecuación matricial representa un conjunto de ecuaciones lineales que se pueden resolver para el vector desconocido de amplitudes, Prony propuso definir el polinomio que tiene los exponentes z_k como sus raíces, la ecuación 9 muestra la expresión definida:

Ecuación 9. Función polinomial en forma de productos

$$F(z) = \prod_{K=1}^{M} (z - z_k) = (z - z_1)(z - z_2) \dots (z - z_M)$$

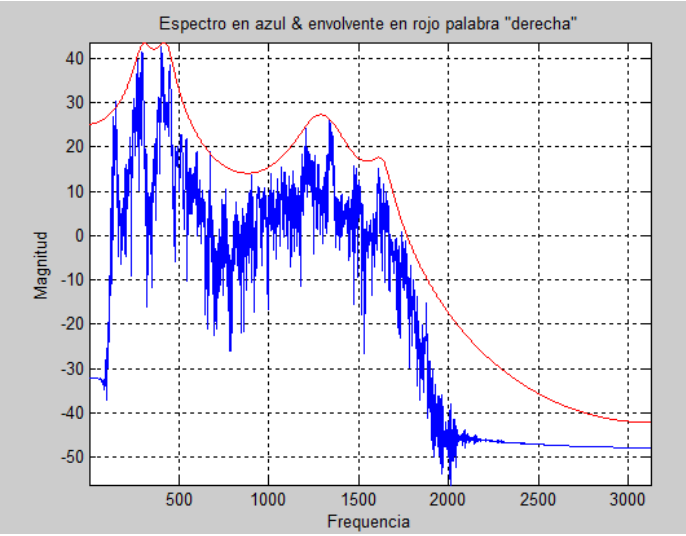
Y en forma de sumatoria es representada como se muestra en la ecuación 10.

Ecuación 10. Función polinomial en forma de Sumatoria

$$F(z) = \sum_{m=0}^{M} a[m] z^{M-m} = a[0]z^M + a[1]z^{M-1} + \cdots + a[M-1]z + a[M]$$

La ecuación se puede resolver para los coeficientes polinomiales. Los factores de amortiguamiento y las frecuencias sinusoidales puede determinarse a partir de las raíces z_k.[13]

Figura 8. Envolvente producida por los coeficientes LPC de la palabra "DERECHA"



Fuente: Autores

La figura 8 se observa la envolvente en el dominio frecuencial de la palabra derecha.

2.1.8 Descomposición de valores singulares

Obteniendo los coeficientes LPC de cada lóbulo se procede a la etapa de reconocimiento de las palabras establecidas calculando los valores singulares (SVD), La teoría muestra que se tiene una matiz A mxn, los valores singulares de A son las raices cuadradas de los autovalores de A^TA, y se denotan mediante $\sigma_1, \dots, \sigma_n$. Es una convención acomodar los valores singulares de modo que $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.

Se puede demostrar matemáticamente que una matriz A de mxn ($m \geq n$) se puede factorizar como se muestra en la ecuación 12.

Ecuación 12. Factorización de la matriz A

$$A = U \sum V^T$$

Donde U es una matriz con columnas ortogonales de mxn, V es una matriz ortogonal de nxn, y Σ una matriz diagonal de nxn. Si los valores singulares no nulos de A son $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, con $r \leq n$ [14].

2.1.9 Distancia Euclidiana

Se utiliza la matriz diagonal S que contiene los valores singulares y se extrae los valores para sacar un promedio de valores entre cada comando y proceder a compararlos con los patrones de referencia previamente almacenados en memoria usando una medida de similitud, la distancia euclidiana permite hallarlo mediante la ecuación 13.

Ecuación 13. Distancia euclidiana

$$d = \sqrt{\sum_{i=1}^{n}(pi - qi)^2}$$

donde d es la distancia, pi es los valores finales de cada comando y qi los valos del usuario obtenido en el momento de dar la accion[15].

La distancia mínima será el comando reconocido por el sistema.

2.2 Software

En primera instancia se implementó el código utilizando Matlab en el que se simuló el proceso de reconocimiento para utilizarlo como base en el desarrollo del software final.

El software se desarrolló utilizando dos lenguajes de programación Python y C, los cuales trabajan de forma alterna. El primero maneja el programa principal el cual se encarga de tomar la señal de voz, llamar el ejecutable que contiene el proceso en C y del control de los micromotores. El segundo realiza el proceso de reconocimiento del comando de voz para ser ejecutado en el sistema.

2.2.1 Python

El programa principal recoge la señal de voz en un periodo de tiempo de dos segundos a una frecuencia de muestreo de 11025 Hz. El audio es almacenado en un formato wave para luego ser convertido en un vector y guardado en un archivo formato de texto, esta conversión se hace para que el programa de reconocimiento implementado en lenguaje C lo pueda leer, procesar y al final entregar un formato igual donde se tiene la información que corresponde al comando reconocido y así después ser leído por Python y realizar la función de control.

Las librerías que usa el código principal son descritas en la Tabla 2.

Tabla 2. Librerías de Python

Libre9ría	Descripción
Pyaudio	Usada principalmente para reproducir y grabar audio.
Wave	Usada para la escritura y lectura del formato wave
Soundfile	Usada para la lectura y escritura de archivos de sonido
Numpy	Encargado de añadir toda la capacidad matemática y vectorial a Python haciendo posible operar con cualquier dato numérico o array
Os	Permite acceder a funcionalidades dependientes del Sistema Operativo.
Gpio	Permite establecer pines de entrada y salida de la raspberry pi.

2.2.2 Lenguaje C

El proceso de reconocimiento y a su vez el más complejo se realizó en este lenguaje dado que el código fuente podría ser muy rápido al ejecutarse.

El algoritmo comienza con la lectura de las muestras del audio que se guardó como texto por un programa en Python.

Las librerías básicas utilizada por este lenguaje se describen en la Tabla 3.

Tabla 3. Librerías básicas del lenguaje C

Librería	Descripción
Stdio.h	Contiene definiciones de macros, constantes, y las declaraciones de funciones para realización de operaciones.
Stdlib.h	Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.
Math.h	Contiene varias funciones matemáticas.

Además, se diseñó las siguientes funciones que son parte del proceso de reconocimiento.

2.2.2.1 Pre filtrado (Pre_filter.c)

Es implementado un filtro digital Butterworth pasa banda de 40 a 1000 Hz de cuarto orden, los coeficientes de este filtro son extraídos por medio de la función butter de Matlab.

La ecuación en diferencia se muestra en la ecuación 14.

Ecuación 14. Ecuación en diferencia del pre filtrado de la señal

$$\sum_{k=0}^{j=N} a_k * y[n-k] = \sum_{k=0}^{j=M} b_k * x[n-k]$$

Donde $\boldsymbol{a_k}$ son los coeficientes de entrada, $\boldsymbol{b_k}$ son los coeficientes de salida[16].

2.2.2.2 Ruido (ruido.c)

Al igual que el anterior se implementa un filtro digital Butterworth pasa banda, de orden 14 con un rango de operación de 300 a 3000 Hz, de esta manera garantiza una mayor eficiencia para su fin que es la eliminación del ruido de la señal. Del mismo modo los coeficientes del filtro son extraídos por la función butter de Matlab y es representado en la Ecuación 14.

2.2.2.3 Preénfasis (enfasis.c)

Se implementa un filtro FIR con un cero cercano a la unidad pasa alto que incremente la energía de las componentes de alta frecuencia que está dada por la Ecuación 15.

Ecuación 15. Filtro preénfasis

$$y(n) = x(n) + x(n-1) * 0.95$$

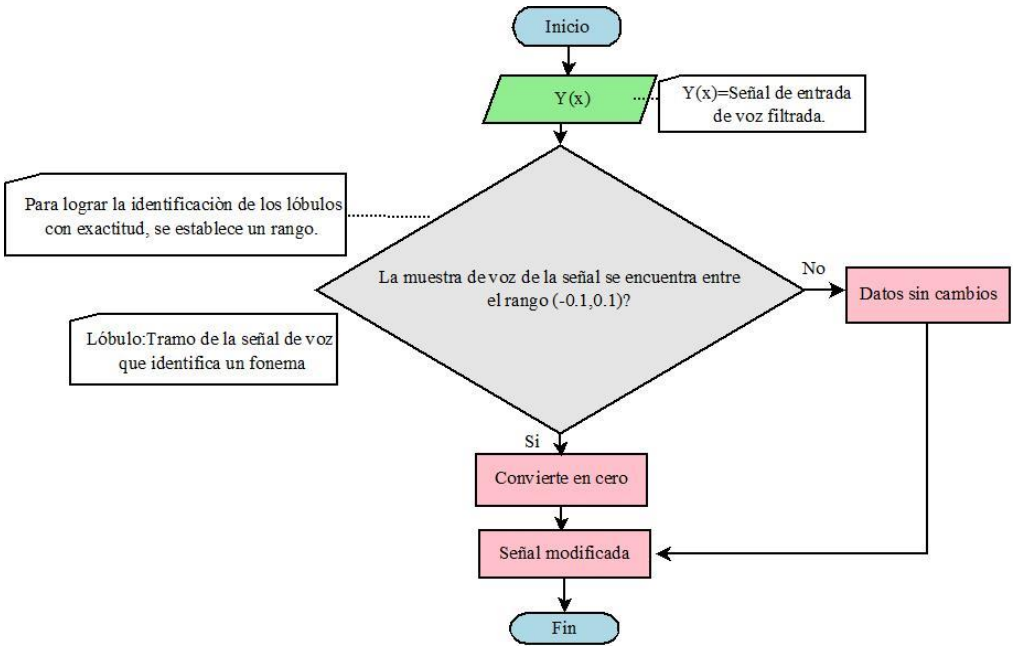
2.2.2.4 Normalización (normalizar.c)

Con base del array resultante del proceso anterior, se extrae el valor mayor y se divide por cada uno de los elementos, así como se muestra en la Ecuación 3.

2.2.2.5 Divisor de audio (divisor_audio.c)

Se convierte a todos los valores que estén entre -0.1 y 0.1 en 0, de esta manera se logra distinguir los lóbulos de la señal de voz. La Figura 9 muestra el diagrama de flujo del proceso.

Figura 9. Diagrama de flujo de la función divisor de audio



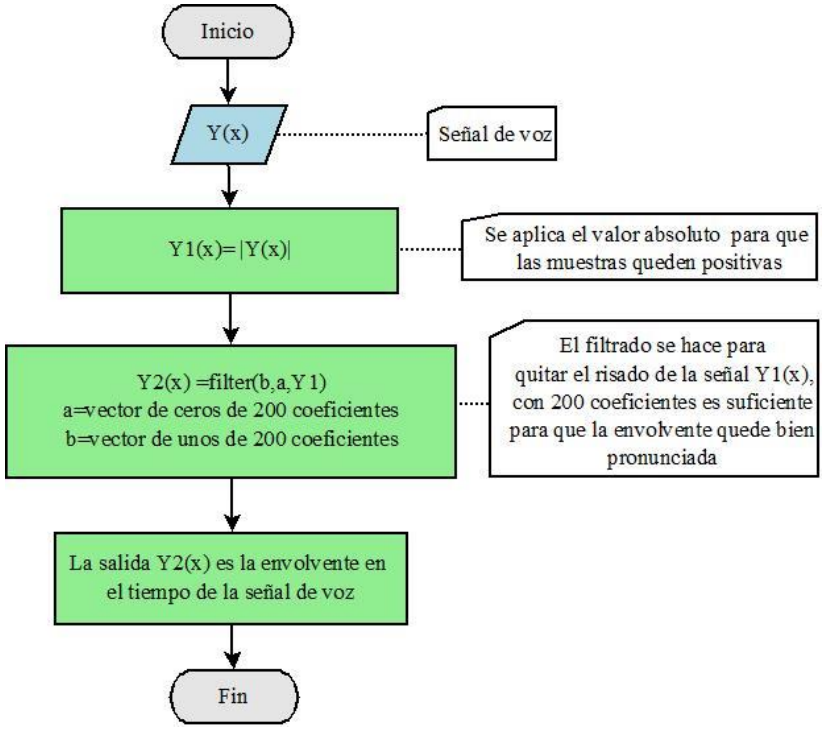
Fuente: Autores

2.2.2.6 Energía (energia.c)

Se calcula un nuevo valor a partir de un grupo de muestras anteriores para generar una envolvente de la señal de audio en el tiempo.

En la Figura 10 se muestra el diagrama flujo de la función energía.

Figura 10. Diagrama de flujo de la función energía



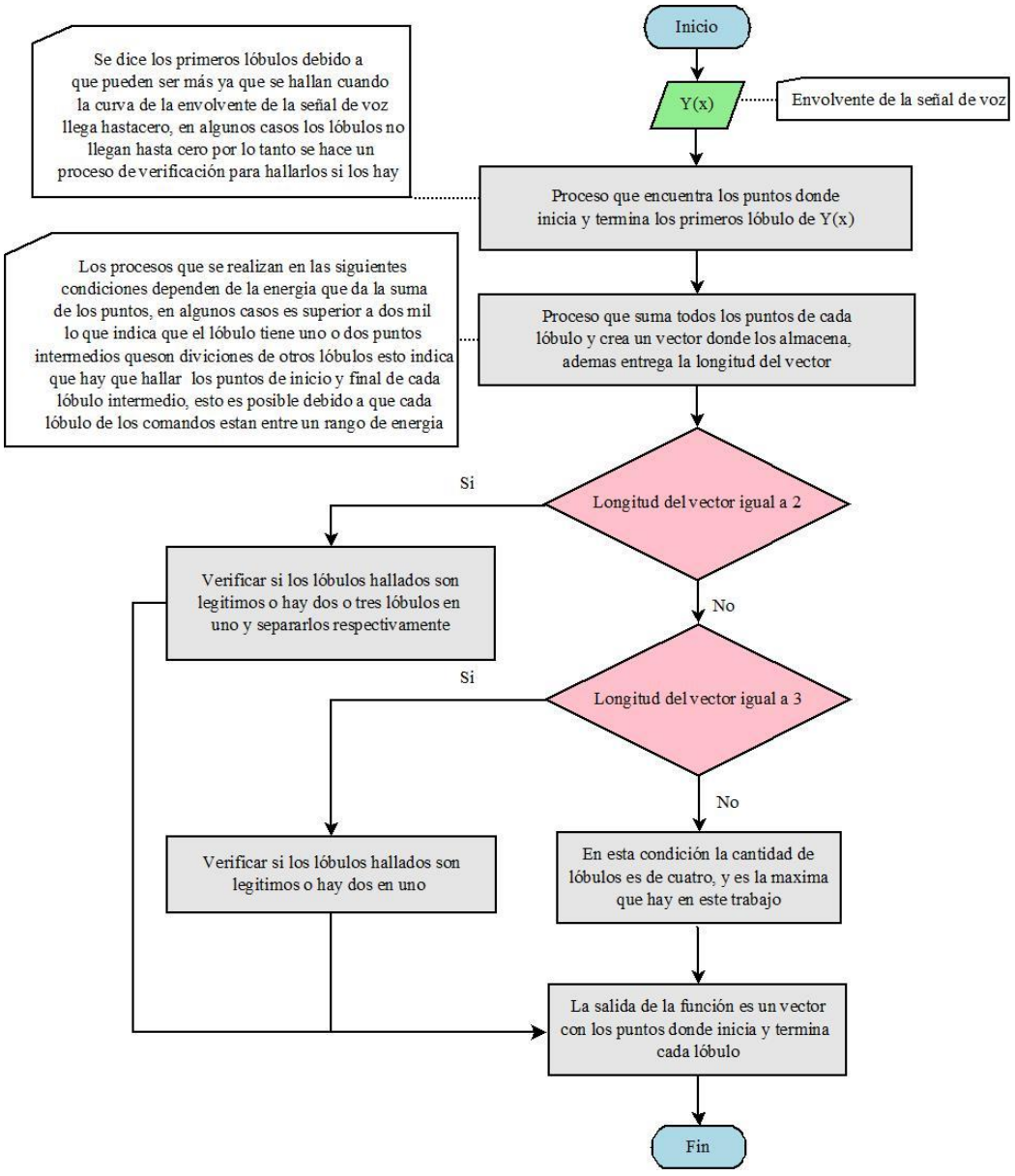
Fuente: Autores

2.2.2.7 divisor de energía (divisor_energía.c)

Esta función consta de una serie de condiciones que permite extraer los rangos entre los que se encuentra cada lóbulo de la señal.

La energía extraída de la función divisor de audio tiene los lóbulos divididos por espacios con valores en cero, esto permite extraer los rangos de los lóbulos iniciales, seguidamente se hace la sumatoria de cada lóbulo para eliminar los más pequeños que pueden ser posibles ruidos, después se determina una serie de condiciones que incluye el valor de energía y de lóbulos de la señal, de esta manera se precisa si los lóbulos iniciales están compuestos por más de uno. En la Figura 11 muestra el diagrama de flujo de la función de la función divisor de energía.

Figura 11. Diagrama de flujo de la función divisor de energía

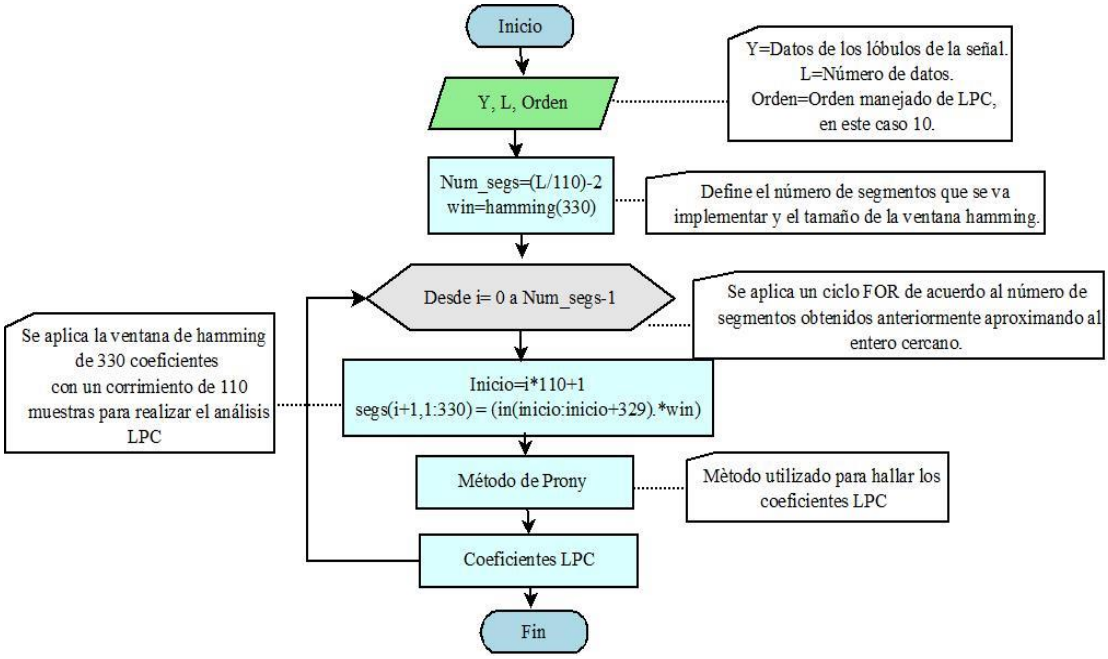


Fuente: Autores

2.2.2.8 Ventana de Hamming y coeficientes LPC (Hamming_LPC.c)

La implementación del algoritmo es con base del diagrama de flujo mostrado en la Figura 12.

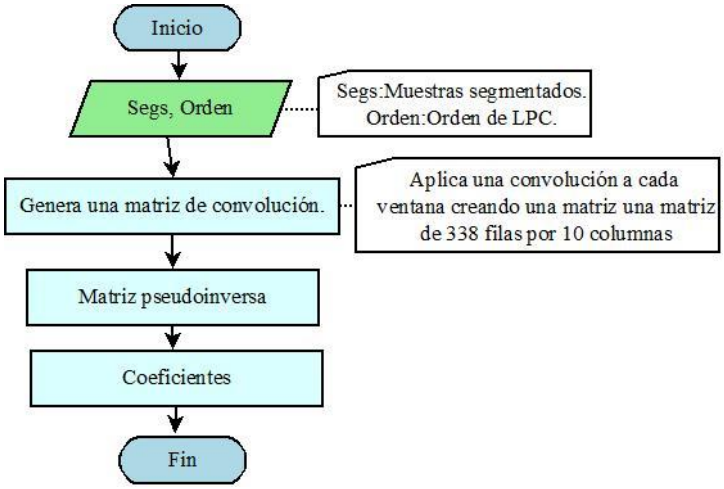
Figura 12. Diagrama de flujo de la función Hamming y coeficientes LPC



Fuente: Autores

El vector de coeficientes de predicción lineal se calcula mediante el algoritmo implementado con el método de Prony, dicha función permite hallar la pseudoinversa de una matriz y para ello se apoya del algoritmo SVD. En la Figura 13 muestra el diagrama de flujo del proceso en general del algoritmo de Prony.

Figura 13. Diagrama de flujo de la función método de Prony



Fuente: Autores

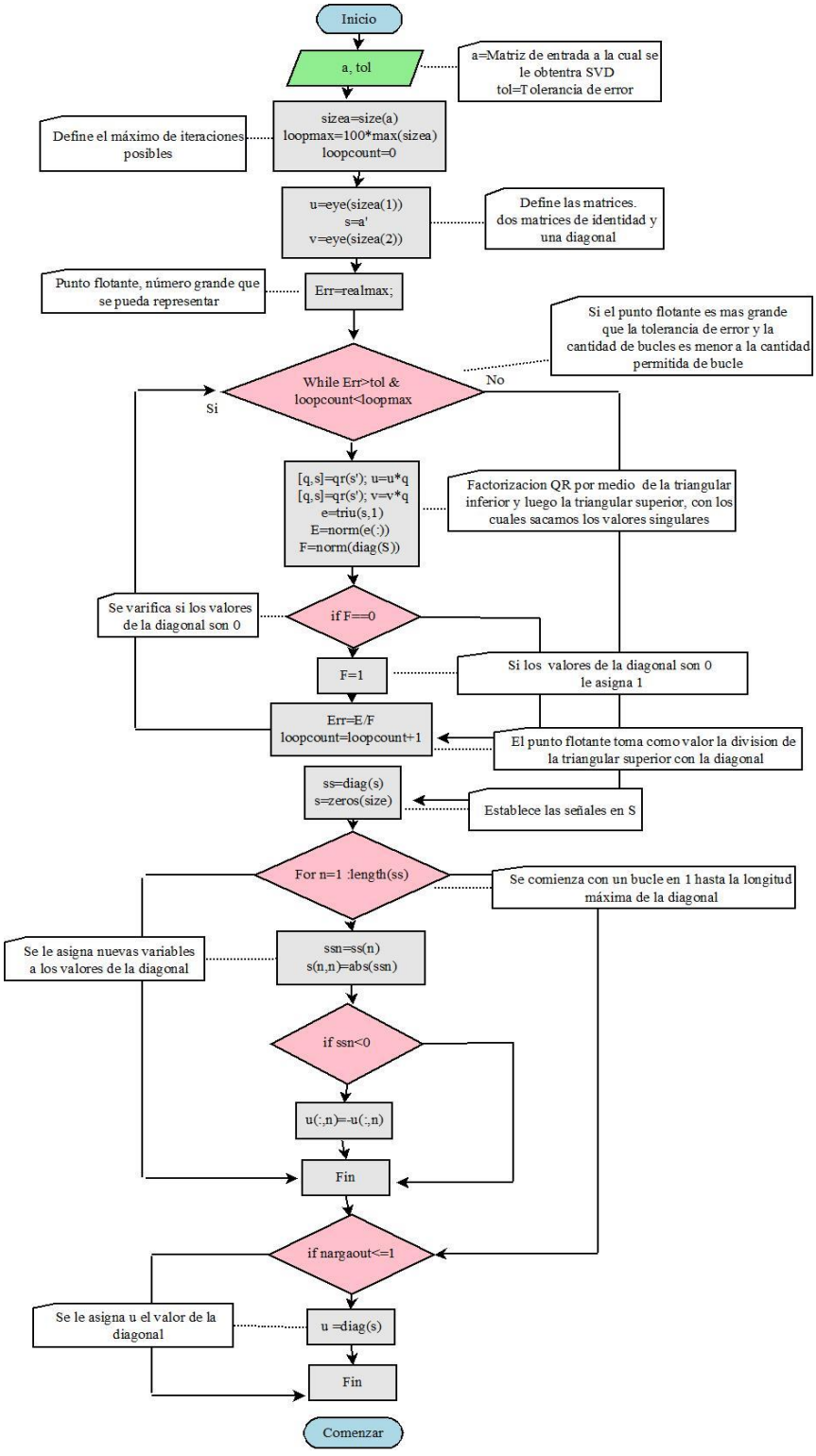
2.2.2.9 SVD (svd.c)

El algoritmo SVD aplica la trasformación Householder para reducir la matriz hermitiana y de esta manera hallar los valores y vectores propios de una matriz (QR), a partir de estos se halla la descomposición en valores singulares[17].

El método LPC se realiza a cada ventana hamming aplicada a la señal de voz para obtenerlos predictores que corresponden al orden del LPC, en este caso 10 coeficientes ,por cada lóbulo de la señal de voz identificado se tendrá una matriz de coeficientes LPC, como esta matriz no tiene un numero definido de filas hace que la aplicación de la distancia euclidiana sea imposible por lo tanto se aplica nuevamente la función SVD, y se toma como resultado la matriz diagonal de valores singulares que tiene un número determinado de muestras.

La función SVD en Matlab fue soporte de apoyo para su implementación, la Figura 14 muestra el diagrama de flujo del proceso paso a paso.

Figura 14. Diagrama de flujo de la función SVD



Fuente: http://www.jovenesenlaciencia.ugto.mx/index.php/jovenesenlaciencia/article/view/2684

2.2.2.10 Interpolación (interp1.c)

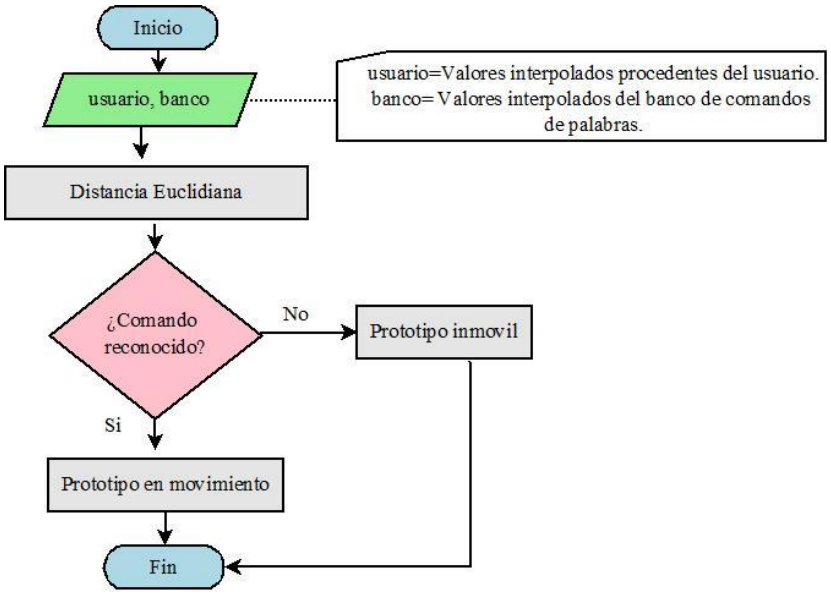
Esta función devuelve valores interpolados de los datos obtenidos de la función Hamming_LPC.c de una sola dimensión usando interpolación lineal, de esta manera, asegura una aproximación de las curvas de los patrones y tener una mejor correlación de los datos.

2.2.2.11 Distancia euclidiana (distancia.c)

Esta función hace una relación punto a punto del vector entregado por la función Interp1 y los vectores del banco de palabras entregando la correlación con cada uno y tomando como resultado la más baja. Si la correlación es menor que 10 el resultado será la posición de la correlación que se encuentra entre 1-5, de lo contrario será cero que indica que el comando no fue reconocido.

La Figura 15 muestra el diagrama de flujo del proceso de clasificación mediante el método de la distancia euclidiana.

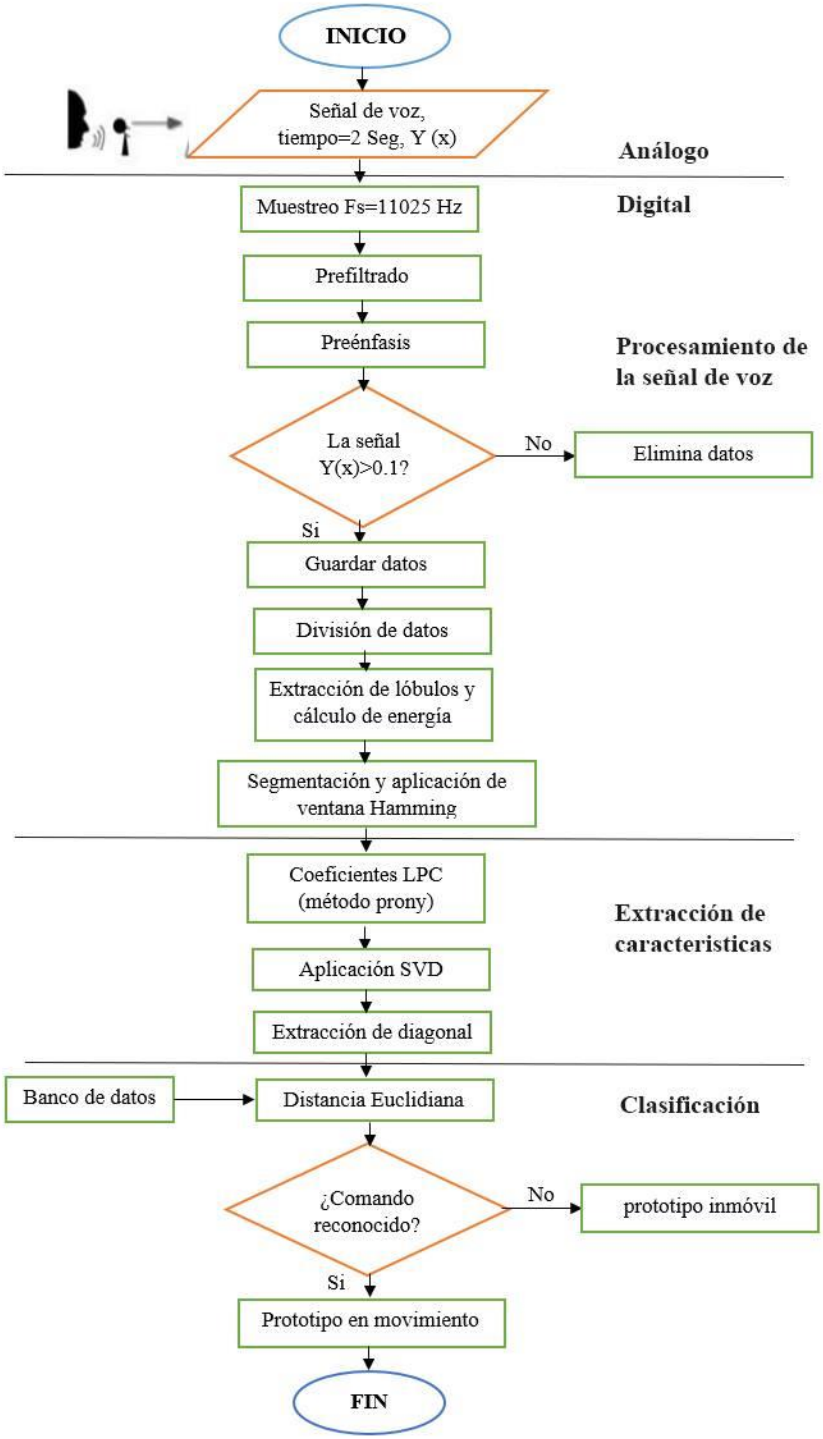
Figura 15. Diagrama de flujo del proceso de clasificación



Fuente: Autores

La figura 16 muestra el diagrama de flujo del proceso de reconocimiento de palabras mediante el método LPC y los pasos expuestos anteriormente.

Figura 16. Diagrama de flujo para el reconocimiento de palabras



Fuente: Autores

2.2.3 Control de los servomotores

El control se realiza desde el programa principal el cual está implementado en Python. El proceso comienza leyendo la información guardada en formato texto, que indica que acción realizar, la información que se encuentra allí es un número del cero al cinco, donde cero indica que no se reconoció el comando, y de uno a cinco corresponde a un comando de movimiento como se muestra en la siguiente tabla:

Tabla 4. Comandos

Numero	Comando
0	No reconocido
1	Izquierda
2	Derecha
3	Atrás
4	Adelante
5	Alto

Cada acción está asociada a cuatro pines de la raspberry pi excepto cuando no se reconoce el comando, dos corresponden al motor derecho y los otros dos al motor izquierdo. Estos pines están conectados a un driver del cual se conectan los motores. La lógica se basa en poner en alto o bajo cada salida dependiendo del comando requerido por el usuario y haciendo que el prototipo realice la acción pedida. Los pines 11 y 12 son los encargados de controlar el movimiento del motor izquierdo, y los pines 13 y 15 el motor derecho.

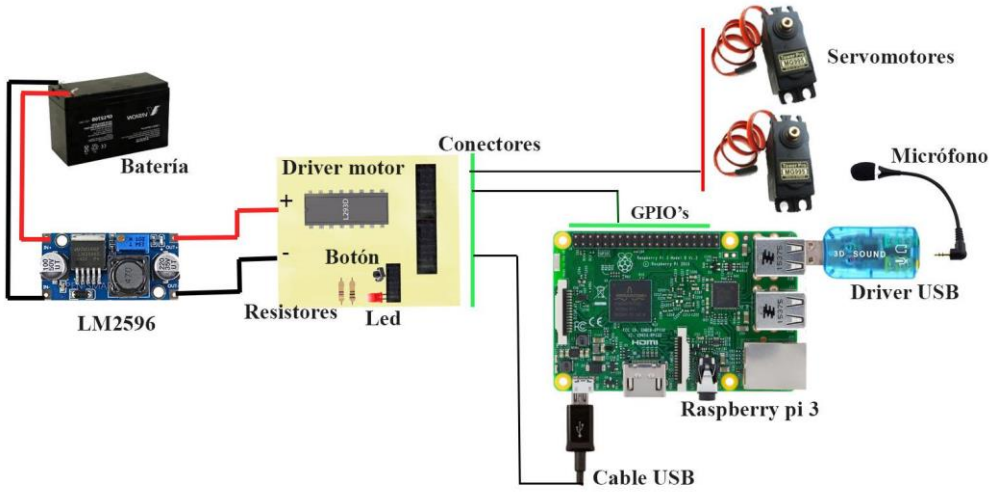
El funcionamiento del prototipo esta hecho de la siguiente manera:

- Si la orden requerida es adelante, el prototipo avanzara de forma permanente hasta que una nueva orden sea reconocida.
- Si la orden requerida es atrás, el prototipo retrocederá de forma permanente hasta que una nueva orden sea reconocida.
- Si la orden requerida es derecha y el prototipo está en movimiento el motor derecho se detendrá por un segundo cambiando de dirección, pero si está detenido se accionará el motor izquierdo por el mismo tiempo.
- Si la orden requerida es izquierda y el prototipo está en movimiento el motor izquierdo se detendrá por un segundo cambiando de dirección, pero si está detenido se accionará el motor derecho por el mismo tiempo
- Si la orden requerida es alta y el prototipo está en movimiento se detendrá.

2.3 Componentes de Hardware

Los componentes del módulo de reconocimiento de palabras se muestran en la Figura 17.

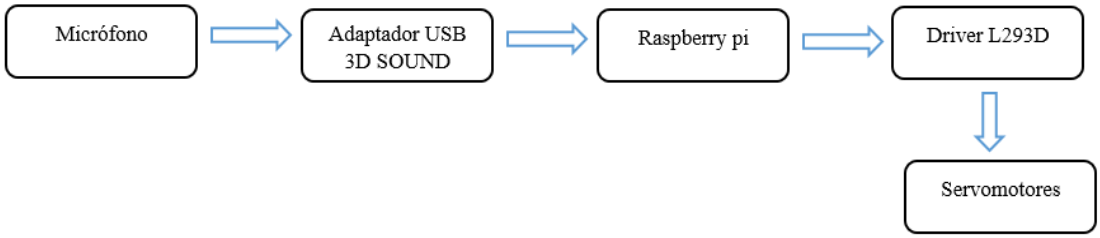
Figura 17. Sistema del reconocimiento de palabras



Fuente: Autores

En la Figura 18 se muestra el diagrama de bloques de los componentes de hardware del sistema.

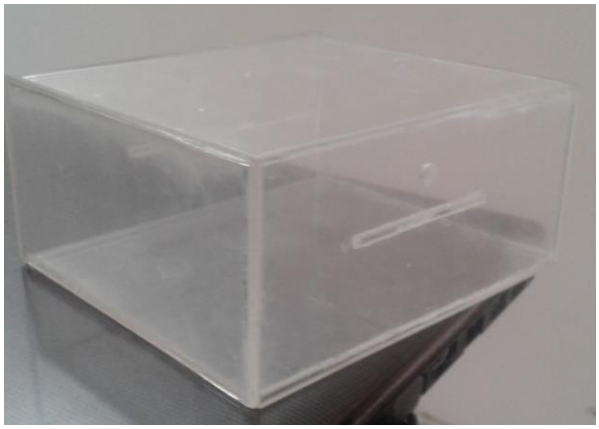
Figura 18. Diagrama de bloques del sistema



Fuente: Autores

Para la protección del sistema se diseña una caja en acrílico con dimensiones de base de 10 x 12 cm y una altura de 5cm.

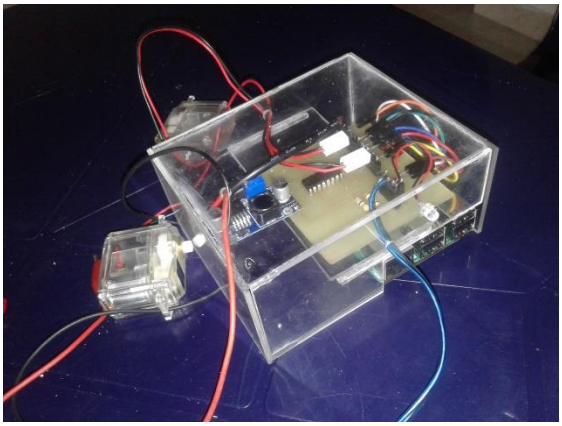
Figura 19. Caja en acrílico



Fuente: Autores

El modulo ya implementado finalmente se muestra en la figura 20.

Figura 20. Módulo de reconocimiento de palabras



Fuente: Autores

En la realización de las pruebas se utiliza un prototipo de silla de ruedas a escala 1:3 adquirida de la universidad Surcolombiana. En la figura 21 se muestra la implementación.

Figura 21. Implementación del módulo de reconocimiento de palabras en una silla de ruedas

Fuente: Autores

3. ANALISIS Y RESULTADOS

3.1 Recomendaciones para el manejo del prototipo

A continuación, se exponen los pasos que se deben llevar a cabo para la manipulación del prototipo de silla de ruedas.

- ✓ Verificar que la batería esté conectada al prototipo de silla de ruedas.
- ✓ Activar el interruptor de encendido y esperar 20 segundos para que se inicie el sistema.
- ✓ Presionar el botón que se encuentra al lado del micrófono e inmediatamente se encenderá un led que indica que puede decir el comando de la acción que quiere que se realice.
- ✓ Al momento de pronunciar el comando de voz procure hacerlo de forma pausada y clara, para así tener una mejor respuesta del sistema.
- ✓ El led que indica el momento de pronunciar el comando tiene un tiempo de encendido de dos segundos y cuando se apague se realiza la acción.
- ✓ Si el sistema no responde o responde incorrectamente espere a que el led se apague para presionar el botón y decir el comando de voz nuevamente.

3.2 Pruebas de funcionamiento

En la realización del banco de datos fue necesaria la recolección de audios de 10 hombres y 10 mujeres de diferentes edades y tonos de voz.
Para analizar el rendimiento del sistema se realizaron pruebas en dos ambientes diferentes, uno en un lugar silencioso y otro con ruido. Las siguientes tablas muestran los resultados obtenidos al pronunciar 10 veces cada palabra por hombres y mujeres.

Tabla 5. Resultados obtenidos en un ambiente silencioso

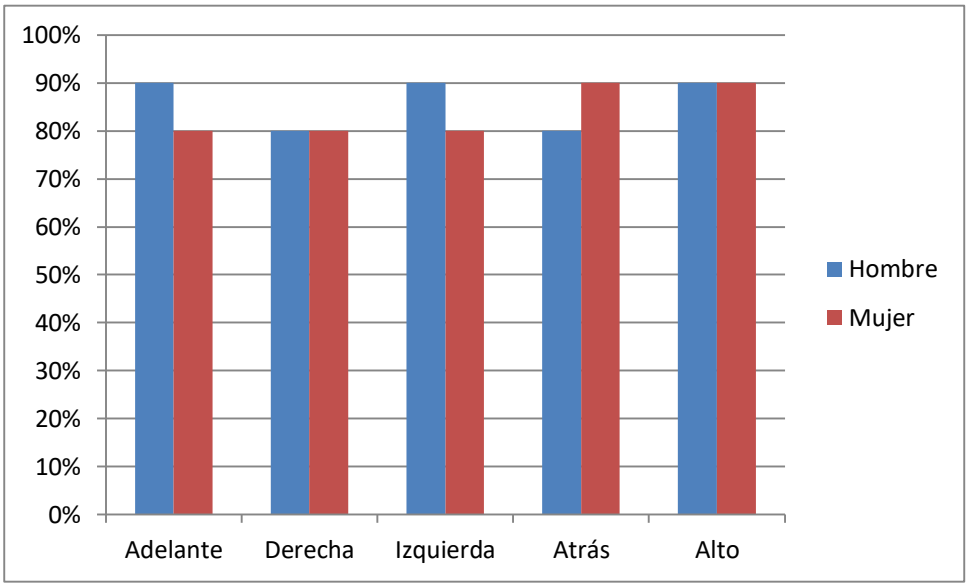
Comando	Hombre		Mujer	
	Acertada	Error	Acertada	Error
Adelante	9	1	8	2
Derecha	8	2	8	2
Izquierda	9	1	8	2
Atrás	8	2	9	1
Alto	9	1	9	1

Tabla 6. Resultados obtenidos en un ambiente ruidoso

Comando	Hombre		Mujer	
	Acertada	Error	Acertada	Error
Adelante	8	2	7	3
Derecha	7	3	8	2
Izquierda	7	3	7	3
Atrás	8	2	7	3
Alto	8	2	8	2

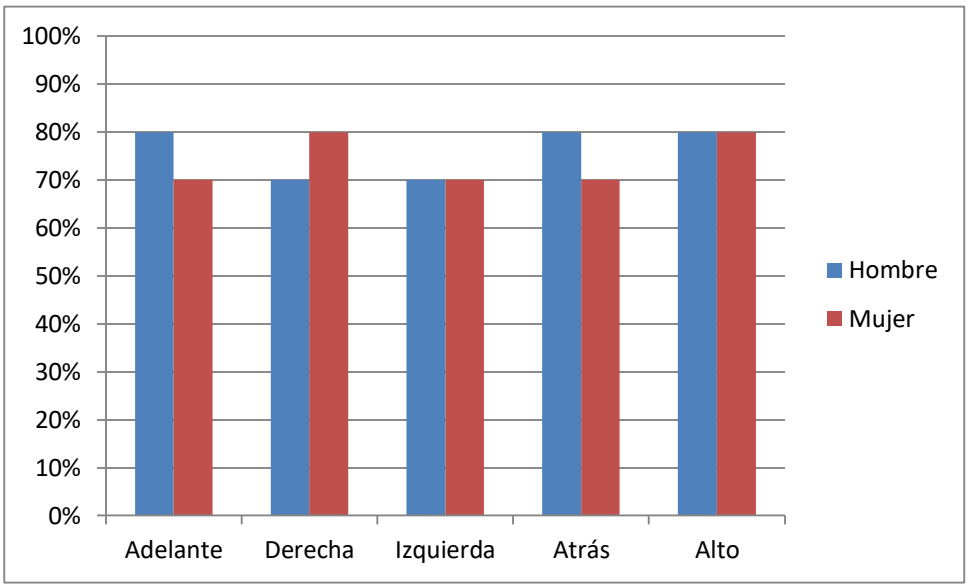
Las siguientes graficas muestran el nivel porcentual de los comandos reconocidos obtenidos en las pruebas realizadas con el prototipo de silla de ruedas en un ambiente silencioso y con ruido externo, con la participación de dos hombres y dos mujeres a los que se les pidió que repitieran cada comando cinco veces.

Grafica 1. Resultados porcentuales obtenidos en un ambiente en silencio



En la Gráfica 1 se muestra los resultados obtenidos en un ambiente silencioso, donde se tiene que el mayor porcentaje es de un 90% y el menor de un 80% y con un promedio en los hombres de un 86% y el en las mujeres un 84% en el que se obtiene un promedio general de un 85%. Estos resultados indican que el sistema es fiable y bueno.

Grafica 2. Resultados porcentuales obtenidos en un ambiente con ruido



En la Gráfica 2 se muestra los resultados obtenidos en un ambiente con ruido, donde se tiene que el mayor porcentaje es de un 80% y el menor de un 70% y con un promedio en los hombres de un 76% y el en las mujeres un 74% en el que se obtiene un promedio general de un 75%. Estos resultados indican que el sistema sigue siendo fiable.

Teniendo en cuenta el comportamiento del sistema en los dos ambientes se puede ver que hay una diferencia del 10% entre los dos, esto se debe al ruido externo que afecta a la señal de voz del comando pronunciado por el usuario.

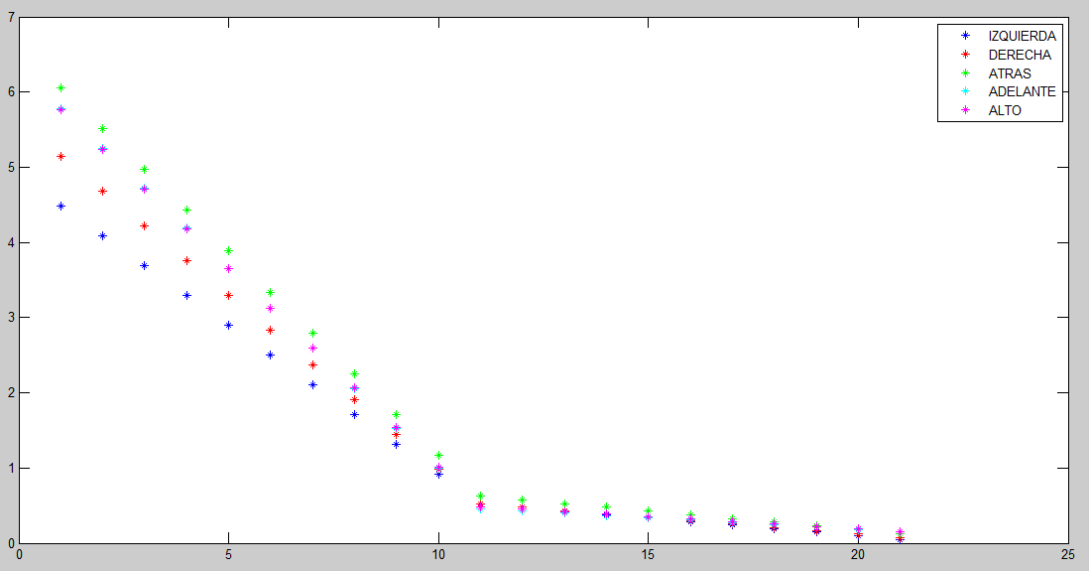
3.3 Métodos utilizados para el mejoramiento del reconocimiento de palabras

Para el mejoramiento del sistema se hicieron dos procesos adicionales a la extracción de los coeficientes de predicción lineal LPC que es la matriz diagonal de valores singulares del SVD y la interpolación de estos valores.
En el proceso de extracción de características, se hizo un análisis por cada lóbulo de la señal de voz, donde se obtuvo como resultado una matriz de coeficientes LPC por cada uno. Esta matriz contiene un número definido de columnas el cual hace referencia a los 10 coeficientes LPC por ventana Hamming que se aplicó y un número indeterminado de filas que depende de la cantidad de desplazamientos de la ventana hasta completar la longitud del lóbulo, esto hace que sea imposible calcular la distancia euclidiana si se tiene en cuenta que los vectores

tienen que ser del mismo tamaño. Por esta razón se calcula la matriz diagonal de valores singulares el cual entrega un vector de diez muestras por cada matriz de LPC, posteriormente se calcula una interpolación lineal para tener una mejor definición de las curvas formadas por el promedio de las diagonales de los valores singulares de cada comando de voz.

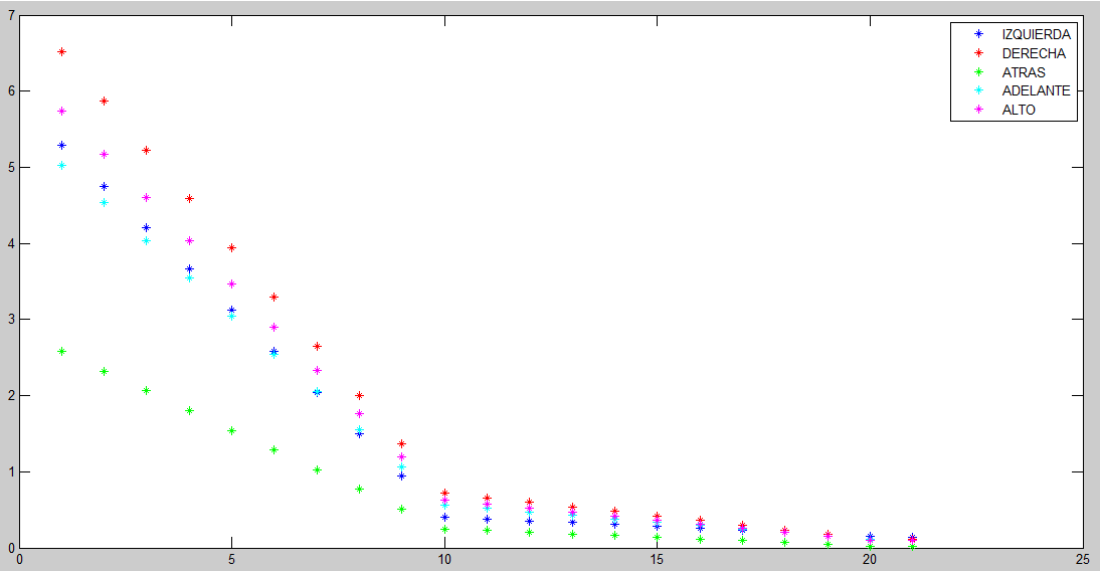
De la figura 24 a 27 se muestran las curvas de los patrones interpolados de cada lóbulo de los comandos de voz del banco de datos, estas figuras corresponden a un solo vector por lo que se tiene que están conectas en el orden que se exponen

Figura 22. Promedio interpolado de los valores singulares del primer lóbulo de cada comando de voz



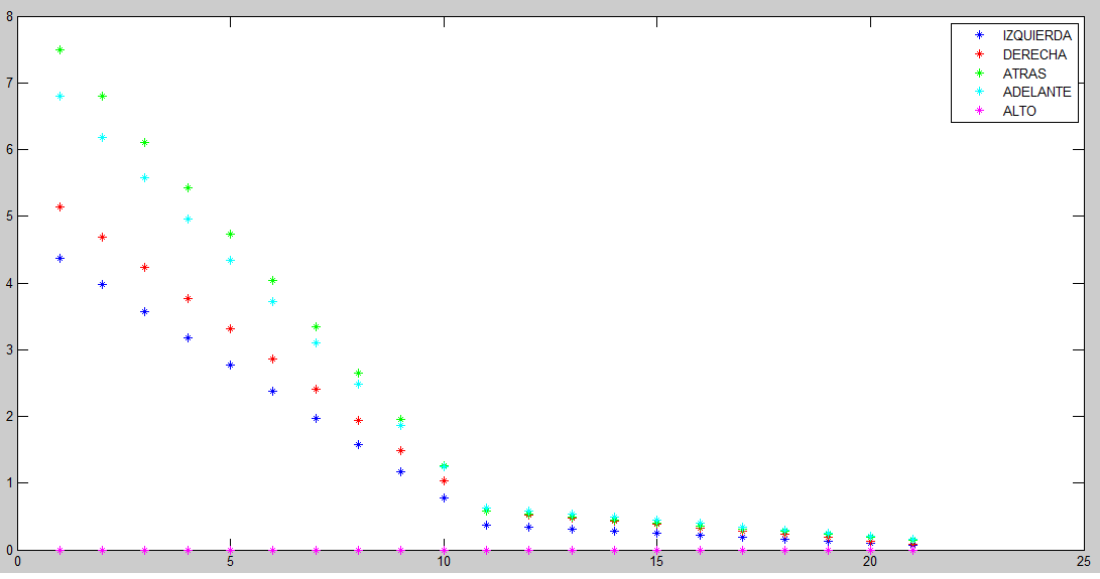
Fuente: Autores

Figura 23. Promedio interpolado de los valores singulares del segundo lóbulo de cada comando de voz



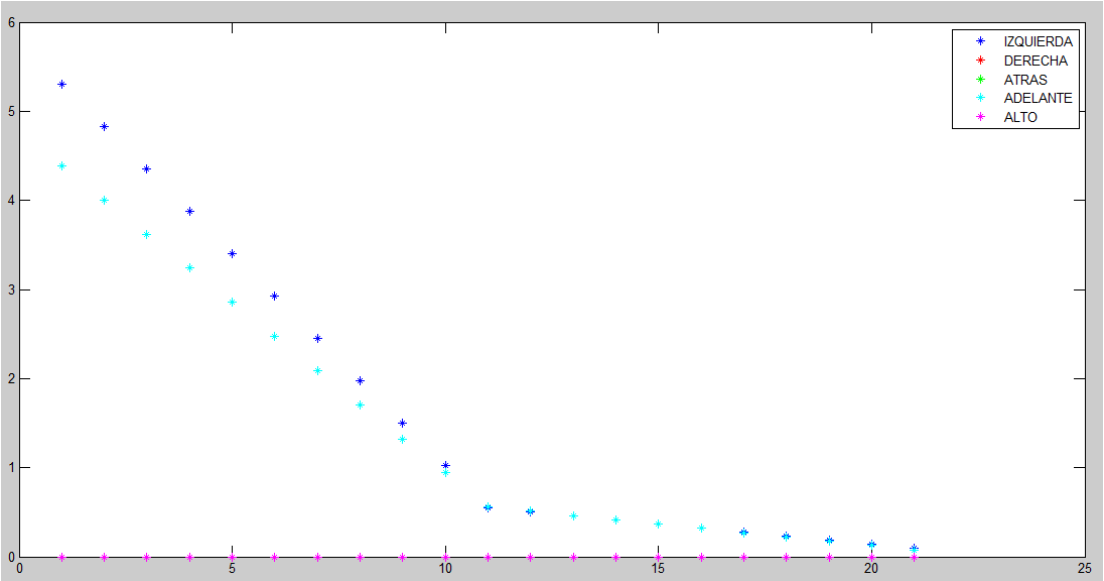
Fuente: Autores

Figura 24. Promedio interpolado de los valores singulares del tercer lóbulo de cada comando de voz



Fuente: Autores

Figura 25. Promedio interpolado de los valores singulares del cuarto lóbulo de cada comando de voz



Fuente: Autores

De la figura 22 a la 25 se muestra los rangos en los que las señales tienen mayor diferencia entre ellas, presentando que en algunas graficas dejan de dibujar ciertas curvas debido a que corresponden a los lóbulos de los comandos de palabras. Llevando este orden de ideas se tiene lo siguiente:

- ✓ Adelante tiene 4 lóbulos.
- ✓ Izquierda tiene 4 lóbulos.
- ✓ Derecha tiene 3 Lóbulos.
- ✓ Atrás tiene 3 lóbulos.
- ✓ Alto tiene 2 lóbulos.

En la Figura 22 y 23 se puede observar que todas las curvas correspondientes a los comandos de voz están presente puesto a que es el primer y segundo lóbulo de todos los comandos, en la Figura 24 la curva del comando alto cae a cero debido a que solo tiene dos lóbulos, en la Figura 25 la curva de los comandos derecha, atrás y alto caen a cero.

4. CONCLUSIONES

- El modelo LPC es un método que permite obtener la información más importante de un contenido extenso de muestras de la señal de voz, de tal manera que el proceso de reconocimiento se desarrolle en menor tiempo y con un alto grado de fiabilidad.

- El desarrollo de un algoritmo de reconocimiento de palabras es una tarea desafiante debido a los diferentes parámetros que se deben tener en cuenta y el procesamiento que se implementa a la señal de voz, para ello se tiene que utilizar técnicas y conceptos que permitan alcanzar una mayor velocidad computacional con un menor gasto de memoria, puesto que esto influye en el rendimiento del sistema.

- La aplicación de la función que divide los comandos de voz por lóbulos mejoro el reconocimiento de patrones debido a que el análisis se hace a cada sección del comando dando como resultado una mejor correlación.

- Para que el sistema de reconocimiento sea robusto fue necesario tomar diferentes muestras de audios de distintas personas para obtener un rango de eficiencia más alto.

- El módulo de reconocimiento de palabras se implementó en el prototipo de silla de ruedas cumpliendo con el objetivo expuesto, además puede ser implementado en otros sistemas donde se requieran estos comandos, también tiene la versatilidad de aumentar o disminuir el número de comandos dependiendo de las acciones que se quieran realizar.

- La computadora de placa simple Raspberry Pi 3 es una buena opción para el desarrollo e implementación del algoritmo del sistema de reconocimiento de palabras, debido a que permite un funcionamiento eficiente en los dos lenguajes de programación utilizados a un tiempo de ejecución rápido.

- La aplicación de la diagonal de los valores singulares del SVD a las matrices resultantes del ventaneo de cada lóbulo, redujo considerablemente la complejidad de los resultados a la hora de realizar la comparación de los patrones del banco de datos con el comando dicho por el usuario ya que permite reducir una matriz extensa a un vector con los parámetros más relevantes.

- Aplicar finalmente interpolación a los datos se obtenía nuevos puntos partiendo de los conocidos, por lo tanto, aseguraba una aproximación de las curvas de los patrones y tener una mejor correlación de los datos.

- El módulo de reconocimiento de palabras se activa por medio de un botón debido a que es un sistema que reconoce más de un tipo de voz, por otro lado, si el sistema no tuviera esta limitación si no que se activara al pronunciase el comando puede ser interrumpida por un tercero.

RECOMENDACIONES

El método LPC y las estrategias para la extracción y el reconocimiento de palabras utilizados en este trabajo es uno de varios modelos existentes, un análisis respecto a los demás es conveniente para la determinación del nivel de rendimiento del sistema en general.

Se recomienda realizar en futuros proyectos el módulo de reconocimiento en un DSPic puesto que requiere un estudio más a fondo de optimización del lenguaje C y la programación de microcontroladores avanzados. Además, conlleva a la disminución del consumo de energía haciendo que el sistema de alimentación sea menos costoso y de un menor tamaño.

Implementar el módulo de reconocimiento de palabras en un sistema donde se pueda utilizar las mismas acciones de control. También se puede hacer cambios en el algoritmo para aumentar o disminuir el número de estos comandos dependiendo de la necesidad.

Para una mejor aplicación del módulo de reconocimiento de palabras se puede aplicar a una silla de ruedas a escala natural haciendo las respectivas modificaciones en la etapa de potencia, la alimentación y actuadores.

Se propone modificar el sistema de cableado del micrófono a un micrófono inalámbrico para una mayor comodidad y apariencia física.

BIBLIOGRAFÍA

[1] WIJOYO, Suryo. WIJOYO, Thiang. Speech Recognition Using Linear Predictive Coding and Artificial Neural Network for Cotrolling Movement of Mobile Robot. Indonesia: Electrical Engineering Department, Petra Chistian University, 2011.

[2] SADAOKI, Furui. 50 Years of Progress in Speech and Speaker Recognition Research. ECTI transactions on Computer and Information Technology (ECTI-CIT), vol 1. 2005.

[3] CASTILLO, Karin. CRUZ, Natalia. ESCOBAR, Mauricio. MEDINA, Estela. Fundamentos de la identificación vocal de hablantes del español de Chile: una mirada fonoaudiológica. Santiago de Chile, Chile: Universidad de Chile, Facultad de Medicina. 2011.

[4] GIL, Juana. Los sonidos del lenguaje. Madrid, España: http://liceu.uab.es/~joaquim/phonetics/fon_anal_acus/fon_acust.html. 1988.

[5] GUZMAN, Marco. Acústica del tracto vocal. Santiago de Chile, Chile.

[6] BOCCO, Federico. GIANA, Francisco. RAMOS, Paulo. Procesadores de audio: filtros, generalidades. Argentina, Universidad Tecnológica Nacional, Facultad regional Cordoba. 2011.

[7].RUEDA, Leticia. Mejoras en reconocimiento del habla basadas en mejoras en la parametrización de la voz. España, Universidad Autónoma de Madrid, 2011.

[8] CORINTIOS, Michael. Signals, Systems, Transforms, and Digital Signal Processing with MATLAB. Chapter 12: Energy and Power Spectral Densities. Taylor & Francis Group, LLC. 2009. ISBN-13: 978-1420090482.

[9] MORANTE, Santiago. Precauciones a la hora de normalizar datos en Data Science. Synergic partners. Revisado 3 de septiembre del 2018. Disponible en internet: http://www.synergicpartners.com/precauciones-a-la-hora-de-normalizar-datos-en-data-science/.

[10] SAN MARTIN, Cesar. CARRILLO, Roberto. Implementación de un reconocedor de palabras aisladas independiente del locutor. Chile: Revista facultad de ingeniería. 2004.

[11] POULARIKAS, Alexander, D. The Handbook of Formulas and Tables for Signal Processing. Revisado el 23 de Agosto de 2018. Disponible en internet: http://dsp-book.narod.ru/HFTSP/8579ch07.pdf.

[12] BAQUERO, Yeison. ALEZONES, Zuleica. BORRERO, Henry. Robot móvil controlado por comandos de voz LPC DTW.Villavicencio, Colombia: Universidad de los Llanos. 2011.

[13] TADEUSZ, Lobos. JACEK, Rezmer. Spectral Estimation of Distorted Signals Using Prony Method.

[14] DEMMEL, James. Applied Numerical Linear Algebra, SIAM, Philadelphia. 1997. ISBN: 9780898713893.

[15] BEDOLLA, Jorge A. Aplicación de distancias entre términos para datos planos y jerárquicos. Valencia, España. Universidad Politécnica de Valencia. 2011.

[16] Ecuaciones en diferencia finitas. Revisado el 20 de septiembre del 2018. Disponible en internet: http://www.ehu.eus/Procesadodesenales/tema2/t65.html.

[17] ZABALLA, Ion. Análisis Matricial Aplicado y Ampliación de Métodos Numéricos. España. Universidad del Pais Vasco, Euskal Herriko Unibertsitatea.

ANEXOS

Anexo A. Algoritmo principal en Python

audio.py

```
import pyaudio
import wave
import soundfile as sf
import numpy as np
import os
import RPi.GPIO as GPIO
import time



GPIO.setmode(GPIO.BOARD)
GPIO.setup(3,GPIO.IN)
GPIO.setup(5,GPIO.IN)
GPIO.setup(7,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
GPIO.setup(16,GPIO.OUT)

GPIO.output(13,GPIO.LOW)
GPIO.output(11,GPIO.LOW)
GPIO.output(15,GPIO.LOW)
GPIO.output(12,GPIO.LOW)
s=0
while 1:
    if GPIO.input(3):
        GPIO.output(7,GPIO.HIGH)

        CHUNK = 1024
        FORMAT = pyaudio.paInt16
        CHANNELS = 1
        RATE = 11025
        RECORD_SECONDS = 2
        WAVE_OUTPUT_FILENAME = "/home/pi/MiProye/output.wav"

        p = pyaudio.PyAudio()

        stream = p.open(format=FORMAT,
                        channels=CHANNELS,
                        rate=RATE,
                        input_device_index=1,
                        input=True,
                        output=False,
                        frames_per_buffer=CHUNK)
```

```
print("* recording")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)
print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
data,samplerate=sf.read('/home/pi/MiProye/output.wav')
#json.dump(data,open('dato.txt','wb'))

r=len(data)
#dat=np.array(data)
#with open ('datos.txt','wb')as f:
 #   for line in dat:
np.savetxt('/home/pi/MiProye/datos.txt',data,fmt='%.8f')
print(r)
GPIO.output(7,GPIO.LOW)

os.system("cd /home/pi/MiProye ; ./mm" )
time.sleep(0.1)
x=np.loadtxt('/home/pi/MiProye/numero.txt')
print(x)

if x==1:
    if s==0:
        GPIO.output(13,GPIO.HIGH)
        time.sleep(1)
        GPIO.output(13,GPIO.LOW)
    if s==1:
        GPIO.output(11,GPIO.LOW)
        time.sleep(1)
        GPIO.output(11,GPIO.HIGH)
    if s==2:
        GPIO.output(12,GPIO.LOW)
        time.sleep(1)
        GPIO.output(12,GPIO.HIGH)
if x==2:
    if s==0:
        GPIO.output(11,GPIO.HIGH)
        time.sleep(1)
        GPIO.output(11,GPIO.LOW)
```

```
            if s==1:
                GPIO.output(13,GPIO.LOW)
                time.sleep(1)
                GPIO.output(13,GPIO.HIGH)
            if s==2:
                GPIO.output(15,GPIO.LOW)
                time.sleep(1)
                GPIO.output(15,GPIO.HIGH)
        if x==3:
            GPIO.output(15,GPIO.HIGH)
            GPIO.output(12,GPIO.HIGH)
            GPIO.output(13,GPIO.LOW)
            GPIO.output(11,GPIO.LOW)
            s=2
        if x==4:
            GPIO.output(13,GPIO.HIGH)
            GPIO.output(11,GPIO.HIGH)
            GPIO.output(15,GPIO.LOW)
            GPIO.output(12,GPIO.LOW)
            s=1
        if x==5:
            GPIO.output(13,GPIO.LOW)
            GPIO.output(11,GPIO.LOW)
            GPIO.output(15,GPIO.LOW)
            GPIO.output(12,GPIO.LOW)
            s=0
```

Anexo B. Algoritmo de reconocimiento de palabras en lenguaje C

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pre_filter.h"
#include "ruido.h"
#include "enfasis.h"
#include "normalizar.h"
#include "divisor_audio.h"
#include "energia.h"
#include "normalizar1.h"
#include "divisor_energia.h"
#include "hamming_LPC.h"
#include "ordenar.h"
#include "interp1.h"
#include "distancia.h"

int main(int argc, char *argv[]){

        double ss,*s,*s1,*s2,*s3,*s4,*s5,*s6,*s8,*s9,*s10,*s11,*s13;
```

```
int ss1,i,i1,i2,i4,i5,j,ww,ww1,n,n1=0,n2=0,n3,n4,xx,*s7,i10,i11,e;
n=20480,n1=4000,n2=4000;
n3=n-n1-n2;
double y[n3];
double pInData[20480];
double h1[2500];
double h[5][431];
double S[44];
double x1[44];
double x2[440];
double *yy=(double *) calloc(440,sizeof(double));
int www;
FILE *f3=fopen("/home/pi/MiProye/datos.txt","r");
for(www=0;www<20480;www++){
        fscanf(f3,"%lf ",&pInData[www]);

}
fclose(f3);
for (i=n1;i<=n-n2;i++){
        y[i-n1]=pInData[i];
}

s=pre_filter(y,n3);
s1=ruido(s,n3);
s2=enfasis(s1,n3);
s3=normalizar(s2,n3);
s4=divisor_audio(s3,0.1,n3);
s5=energia(s4,n3);
s6=normalizar1(s5,n3);
s7=divisor_energia(s6,n3);
for(j=0;j<=8;j++){
        if(s7[j]==0){
                break;
        }
}

s8=energia(s2,n3);
s9=normalizar(s8,n3);
xx=0;
for(i5=0;i5<j;i5+=2){
        n4=s7[i5+1]-s7[i5]+1;
        for(i4=0;i4<=s7[i5+1]-s7[i5];i4++){
                h1[i4]=s9[s7[i5]+i4];
        }
        s10=hamming(h1,n4);
        s11=ordenar(s10);
```

```
                for(i2=0;i2<11;i2++){
                        S[i2+xx]=s11[i2];
                }
                xx=xx+11;
        }
        for(i10=0;i10<44;i10++){
                x1[i10]=i10+1;
        }
        for(i11=0;i11<440;i11++){
                x2[i11]=i11*0.1+1;
        }
        interp1(x1,44,S,x2,440,yy);
        FILE *f2=fopen("/home/pi/MiProye/file2.txt","r");
        for(ww=0;ww<5;ww++){
                for(ww1=0;ww1<431;ww1++){
                        fscanf(f2,"%lf ",&h[ww][ww1]);

                }

        }
        fclose(f2);
        s13=distancia(yy,h);
        ss=1000;
        ss1=0;
        for(i1=0;i1<5;i1++){
                if(s13[i1]<ss){
                        ss=s13[i1];
                        ss1=i1+1;
                }
        }
        if(ss<10){
                e=ss1;
        }else{
                e=0;
        }
        FILE *f=fopen("/home/pi/MiProye/numero.txt","w");
                fprintf(f,"%d ",e);
        fclose(f);

        return 0;
}
```

pre_filter.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>
#include "pre_filter.h"

double* pre_filter(double y[],int n){
	static double y1[22050];
	double h[n],a1,*p;
	int i,j,k,a;
	for (i=0;i<n;i++){
		if(y[i]>0){
			h[i]=y[i];
		}else{
			h[i]=y[i]*(-1);
		}
	}
	for(j=0;j<n;j++){
		if(h[j]>a1){
			a1=h[j];
		}
	}
	for (k=0;k<n;k++){
		y[k]=y[k]/a1;
	}
	for(a=0;a<n;a++){
		if (a>=4){
			y1[a]=0.053356372784167*y[a]-0.106712745568334*y[a-
2]+0.053356372784167*y[a-4]+3.227100625566485*y1[a-1]-3.923629031508764*y1[a-
2]+2.158556586031919*y1[a-3]-0.462148262986001*y1[a-4];
		}else if(a>=3){
			y1[a]=0.053356372784167*y[a]-0.106712745568334*y[a-
2]+3.227100625566485*y1[a-1]-3.923629031508764*y1[a-2]+2.158556586031919*y1[a-3];
		}else if(a>=2){
			y1[a]=0.053356372784167*y[a]-0.106712745568334*y[a-
2]+3.227100625566485*y1[a-1]-3.923629031508764*y1[a-2];
		}else if(a>=1){
			y1[a]=0.053356372784167*y[a]+3.227100625566485*y1[a-1];
		}else{
			y1[a]=0.053356372784167*y[a];
		}
/*		printf("%0.4f ",y1[a]);	*/
	}
	p=y1;
	return p;
}
```

Ruido.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "ruido.h"

double* ruido(double y[],int n){
	static double y1[22050];
	double *p;
	int i;
	for(i=0;i<n;i++){
		if (i>=14){
			y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-8]-
0.310591145833710*y[i-10]+0.103530381944570*y[i-12]-0.014790054563510*y[i-
14]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
			-30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9]-1.065036815107152*y1[i-10]+0.258429977500429*y1[i-11]-
0.048053676606371*y1[i-12]+0.007147739456597*y1[i-13]
			-0.000123930414944*y1[i-14];
		}else if (i>=13){
			y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-8]-
0.310591145833710*y[i-10]+0.103530381944570*y[i-12]+5.860824633939581*y1[i-1]-
15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
				-30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9]-1.065036815107152*y1[i-10]+0.258429977500429*y1[i-11]-
0.048053676606371*y1[i-12]+0.007147739456597*y1[i-13];
		}else if (i>=12){
			y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-8]-
0.310591145833710*y[i-10]+0.103530381944570*y[i-12]+5.860824633939581*y1[i-1]-
15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
				-30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9]-1.065036815107152*y1[i-10]+0.258429977500429*y1[i-11]-
0.048053676606371*y1[i-12];
		}else if (i>=11){
			y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-8]-
0.310591145833710*y[i-10]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-
2]+25.350513127582083*y1[i-3]
				-30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9]-1.065036815107152*y1[i-10]+0.258429977500429*y1[i-11];
		}else if (i>=10){
```

```
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-8]-
0.310591145833710*y[i-10]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-
2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9]-1.065036815107152*y1[i-10];
        }else if (i>=9){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-
8]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-
8]+3.169866938683280*y1[i-9];
        }else if (i>=8){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+0.517651909722851*y[i-
8]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7]-7.467017339140101*y1[i-8];
        }else if (i>=7){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+5.860824633939581*y1[i-1]-
15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6]+14.591900031031120*y1[i-7];
        }else if (i>=6){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]-0.517651909722851*y[i-6]+5.860824633939581*y1[i-1]-
15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5]-
23.103487019505614*y1[i-6];
        }else if (i>=5){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-
2]+25.350513127582083*y1[i-3]
                        -30.395244147146823*y1[i-4]+29.322126403198453*y1[i-5];
        }else if (i>=4){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+0.310591145833710*y[i-4]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-
2]+25.350513127582083*y1[i-3]-30.395244147146823*y1[i-4];
        }else if (i>=3){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-2]+25.350513127582083*y1[i-3];
        }else if (i>=2){
                y1[i]=0.014790054563510*y[i]-0.103530381944570*y[i-
2]+5.860824633939581*y1[i-1]-15.481860067510368*y1[i-2];
```

```
                }else if (i>=1){
                        y1[i]=0.014790054563510*y[i]+5.860824633939581*y1[i-1];
                }else{
                        y1[i]=0.014790054563510*y[i];
                }
        }
        p=y1;
return p;
}
```

Énfasis.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "enfasis.h"

double* enfasis(double y1[],int n){
        double *p;
        int i;
        static double y2[22050];
        for(i=0;i<n;i++){
                if (i>=1){
                        y2[i] = y1[i]+0.95*y1[i-1];
                }else{
                        y2[i] = y1[i];
                }
        }
        p=y2;
        return p;
}
```

Normalizar.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "normalizar.h"

double* normalizar(double y[],int n){
        double h[n];
```

```
        double *p1;
        double a=0;
        static double y1[22050];
        int i,j,k;
        for (i=0;i<n;i++){
                if(y[i]>=0){
                        h[i]=y[i];
                }else{
                        h[i]=y[i]*(-1);
                }
        }
        for(j=0;j<n;j++){
                if(h[j]>a){
                        a=h[j];
                }
        }
        for (k=0;k<n;k++){
        y1[k]=y[k]/a;
        }
        p1=y1;
   return p1;
}
```

Divisor_audio.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "divisor_audio.h"

double* divisor_audio(double x[],double s,int n){
        double x1[n],*p;
        static double y[22050];
        int i,j;
        for (i=0;i<n;i++){
                x1[i]=sqrt(x[i]*x[i]);
        }
        for (j=0;j<n;j++){
                if (x1[j]<0.1){
                        y[j]=0;
                }else{
                        y[j]=x[j];
                }
        }
        p=y;
        return p;
```

```
}
```

Energía.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "energia.h"

double* energia(double y[],int n){
	int i,j;
	static double h[22050];
	double y1[n];
	double *p;
	for(j=0;j<n;j++){
		y1[j]=sqrt(y[j]*y[j]);
	}
	for(i=0;i<n;i++){
		if(i==0){
			h[i]=y1[i];
		}else if (i<200){
			h[i]=h[i-1]+y1[i];
		}else{
			h[i]=h[i-1]+y1[i]-y1[i-200];
		}
	}
	p=h;
	return p;
}
```

Divisor_energia.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "divisor_energia.h"

int* divisor_energia(double e1[],int n){
	static int hh[9];
	double e[n];
	int *p;
	double ss=10,ss1=10,ss2,ss3,ee1,ee13=0;
	int hh1[6];
	int ll[20],s=0;
	double ee22[5],ee3[3];
```

```
	int
i,qq,k1,kk,i1,i2,aa=0,aa1=0,i3,i4,i5,i6,i7,i8,k,ff=0,ff1=0,rr1=0,z11,zz11,sa,h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h1
1,h12,h13,h14,h15,h16,h17,h18,h19,h20,h21,h22,h23,h24;
	for(i2=0;i2<n;i2++){
		if(i2==0){
			e[i2]=0;
		}else if(e1[i2]<0.0000000000001 && e1[i2]>-0.0000000000001){
			e[i2]=0;
		}else if(i2==n-1){
			e[i2]=0;
		}else{
			e[i2]=e1[i2];
		}
	}
	for (i=0;i<n-1;i++){
		if(e[i]==0){
			if(e[i+1]!=0){
				ll[s]=i;
				s+=1;
			}
		}else{
			if(e[i+1]==0){
				ll[s]=i;
				s+=1;
			}
		}
	}
	int tt1[s];
	int tt11[s];
	for (sa=0;sa<s;sa++){
		tt1[sa]=ll[sa];
	}
	for(kk=0;kk<s;kk+=2){
		ee1=0;
		for(i1=tt1[kk];i1<=tt1[kk+1];i1++){
			ee1+=e[i1];
		}
/*		printf("%lf ",ee1);*/
		if(ee1>10){
			tt11[kk-aa]=tt1[kk];
			tt11[kk+1-aa]=tt1[kk+1];
			ee22[aa1]=ee1;
			aa1+=1;
		}else{
			aa+=2;
		}
	}
```

```
	int tt[aa1*2];
	for (z11=0;z11<aa1*2;z11++){
		tt[z11]=tt11[z11];
	}

	double ee2[aa1];
	for (zz11=0;zz11<aa1;zz11++){
		ee2[zz11]=ee22[zz11];
/*		printf("%lf ",ee2[zz11]);*/
	}
	if(aa1==1){
		if(ee2[0]<1100){
		for(h1=0;h1<8;h1++){
			if(h1==0){
				hh[h1]=tt[h1];
			}else if(h1<3){
				hh[h1]=(tt[0]+tt[1])/2+199+h1;
			}else if(h1==3){
				hh[h1]=tt[1];
			}
		}

		}else{
			for(h2=0;h2<8;h2++){
				hh[h2]=(h2+1)*10;
			}
		}
	}else if((tt[2]-tt[1])>1000 && ee2[1]>800 && aa1!=4){
			if(aa1==2){
				for(i3=tt[2]+600;i3<tt[3]-800;i3++){ //halla el valor minimo//
					if(e[i3]<ss){
						ss=e[i3];
					}
				}
				for(k=tt[2]+600;k<=tt[3]-800;k++){
					if(ss==e[k]){
						ff=k;
						break;
					}
				}
				for(h3=0;h3<8;h3++){
					if(h3<3){
						hh[h3]=tt[h3];
					}else if(h3<5){
						hh[h3]=(tt[2]+ff)/2+297+h3;
					}else if(h3<7){
						hh[h3]=ff+h3-5;
```

```
                    }else{
                        hh[h3]=tt[3];
                    }
                }

            }else if(aa1==3){
                for(h4=0;h4<8;h4++){
                    if(h4<3){
                        hh[h4]=tt[h4];
                    }else if(h4<5){
                        hh[h4]=(tt[2]+tt[3])/2+297+h4;
                    }else{
                        hh[h4]=tt[h4-2];
                    }
                }

            }else{
                for(h5=0;h5<8;h5++){
                    hh[h5]=(h5+1)*10;
                }
            }
} else if(aa1==2){
        if(ee2[0]<1100 && ee2[0]>500 && ee2[1]<800 && ee2[1]>100){
            for(h6=0;h6<4;h6++){
                hh[h6]=tt[h6];
            }
        }else if(ee2[0]<1400 && ee2[0]>600 && ee2[1]<100){
            for(h7=0;h7<4;h7++){
                if(h7==0){
                    hh[h7]=tt[h7];
                }else if(h7<3){
                    hh[h7]=(tt[0]+tt[1])/2+299+h7;
                }else if(h7==3){
                    hh[h7]=tt[h7];
                }
            }

        }else if(ee2[0]<1100 && ee2[1]>900){
            for(qq=0;qq<1000;qq+=200){
                if(e[tt[2]+qq]<e[tt[2]+qq+100]){
                    if(e[tt[2]+qq+100]>e[tt[2]+qq+200]){
                        rr1=qq+100;
                        break;
                    }
                }else{
                    rr1=qq;
                    break;
```

```
        }
}
ss=10;
for(i3=tt[2]+rr1;i3<tt[3]-900;i3++){ //halla el valor minimo//
        if(e[i3]<ss){
                ss=e[i3];
        }
}
for(k=tt[2]+rr1;k<=tt[3]-900;k++){
        if(ss==e[k]){
                ff=k;
                break;
        }
}
for(h8=0;h8<6;h8++){
        if(h8<3){
                hh1[h8]=tt[h8];
        }else if(h8<5){
                hh1[h8]=ff+h8-3;
        }else if(h8==5){
                hh1[h8]=tt[3];
        }
}
ss2=0;
for(i4=hh1[2];i4<hh1[3];i4++){
        ss2+=e[i4];
}
if(ss2>250){
        ss3=10;
        for(i5=hh1[2]+800;i5<hh1[3]-800;i5++){
                if(e[i5]<ss3){
                        ss3=e[i5];
                }
        }
        ff1=0;
        for(i6=hh1[2]+800;i6<hh1[3]-800;i6++){
                if(ss3==e[i6]){
                        ff1=i6;
                        break;
                }
        }
        if (ff1!=0){
                for(i7=0;i7<8;i7++){
                        if(i7<3){
                                hh[i7]=hh1[i7];
                        }else if(i7<5){
                                hh[i7]=ff1+i7-3;
```

```
                                        }else{
                                                hh[i7]=hh1[i7];
                                        }
                                }
                        }else{
                                for(i8=0;i8<6;i8++){
                                        hh[i8]=hh1[i8];
                                }
                        }
                }else{
                        for(i8=0;i8<6;i8++){
                                hh[i8]=hh1[i8];
                        }
                }
        }else if(ee2[0]>1200){
                ss=10;
                for(i3=tt[0]+800;i3<tt[1]-1000;i3++){ //halla el valor minimo//
                        if(e[i3]<ss){
                                ss=e[i3];
                        }
                }
                for(k=tt[0]+800;k<=tt[1]-1000;k++){
                        if(ss==e[k]){
                                ff=k;
                                break;
                        }
                }
                for(h9=0;h9<6;h9++){
                        if(h9==0){
                                hh1[h9]=tt[h9];
                        }else if(h9<3){
                                hh1[h9]=ff+h9-1;
                        }else{
                                hh1[h9]=tt[h9-2];
                        }
                }
/*              printf("\n");*/
                int w1=0,y,y1;
                for(y=0;y<5;y+=2){
                        ee13=0;
                        for(y1=hh1[y];y1<=hh1[y+1];y1++){
                                ee13+=e[y1];
                        }
                        ee3[w1]=ee13;
/*                      printf("%lf ",ee3[w1]);*/
                        w1+=1;
                }
```

```
                    if(ee3[0]>800 && ee3[1]<1000){
                            ss1=10;
                            for(i3=hh1[0]+800;i3<hh1[1]-800;i3++){ //halla el valor minimo//
                                    if(e[i3]<ss1){
                                            ss1=e[i3];
                                    }
                            }
                            if(ee3[0]>1100 && ss1<0.6){
                                    for(h10=0;h10<8;h10++){
                                            if(h10==0){
                                                    hh[h10]=hh1[h10];
                                            }else if(h10<3){
                                                    hh[h10]=(hh1[0]+hh1[1])/2+h10-1;
                                            }else{
                                                    hh[h10]=hh1[h10-2];
                                            }
                                    }
                            }else if(hh1[1]-hh1[0]>1000){
                                    for(h10=0;h10<8;h10++){
                                            if(h10==0){
                                                    hh[h10]=hh1[h10];
                                            }else if(h10<3){
                                                    hh[h10]=(hh1[0]+hh1[1])/2+h10-1;
                                            }else{
                                                    hh[h10]=hh1[h10-2];
                                            }
                                    }
                            }else{
                                    for(h11=0;h11<6;h11++){
                                            hh[h11]=hh1[h11];
                                    }

                            }
                    }else if(ee3[0]<1200 && ee3[1]>700){
                            for(i3=hh1[2]+800;i3<hh1[3]-1100;i3++){ //halla el valor minimo//
                                    if(e[i3]<ss1){
                                            ss1=e[i3];
                                    }
                            }
/*                          printf("%lf ",ss1);*/
                            for(k1=hh1[2]+800;k1<=hh1[3]-1100;k1++){
                                    if(ss1==e[k1]){
                                            ff1=k1;
                                            break;
                                    }
                            }
/*                          printf("%d ",ff1);*/
```

```
                        if(ff1!=0 && ss1<0.6){
                                for(h12=0;h12<8;h12++){
                                        if(h12<3){
                                                hh[h12]=hh1[h12];
                                        }else if(h12<5){
                                                hh[h12]=ff1+h12-3;
                                        }else{
                                                hh[h12]=hh1[h12-2];
                                        }
                                }
/*                                      printf("%d ",hh[h12]);*/
                        }else{
                                for(h13=0;h13<6;h13++){
                                        hh[h13]=hh1[h13];
                                }

                        }
                }else{
                        for(h14=0;h14<6;h14++){
                                hh[h14]=hh1[h14];
                        }

                }
        }else{
                for(h15=0;h15<8;h15++){
                        hh[h15]=(h15+1)*10;
                }

        }
}else if(aa1==3){
        if(ee2[0]>300 && ee2[0]<800 && ee2[1]>600 && ee2[1]<1200 && ee2[2]<300){
                for(h16=0;h16<6;h16++){
                        hh[h16]=tt[h16];
                }
        }else if(ee2[1]<300 && ee2[2]>800){
                for(h17=0;h17<6;h17++){
                        hh[h17]=tt[h17];
                }
        }else if(ee2[0]>800){
                for(i3=tt[0]+600;i3<tt[1]-600;i3++){ //halla el valor minimo//
                        if(e[i3]<ss1){
                                ss1=e[i3];
                        }
                }
                if(ss1<0.6){
                        for(h18=0;h18<8;h18++){
                                if(h18==0){
```

```
                                        hh[h18]=tt[h18];
                                    }else if(h18<3){
                                        hh[h18]=(tt[0]+tt[1])/2+h18-1;
                                    }else{
                                        hh[h18]=tt[h18-2];
                                    }
                                }

                            }else{
                                for(h19=0;h19<6;h19++){
                                    hh[h19]=tt[h19];
                                }
                            }
                        }else if(ee2[0]<1000 && ee2[1]>900){
                            ss1=10;
                            for(i3=tt[2]+700;i3<tt[3]-700;i3++){ //halla el valor minimo//
                                if(e[i3]<ss1){
                                    ss1=e[i3];
                                }
                            }
                            for(k1=tt[2]+700;k1<=tt[3]-700;k1++){
                                if(ss1==e[k1]){
                                    ff1=k1;
                                    break;
                                }
                            }
                            if(ff1!=0 && ss1<0.6){
                                for(h20=0;h20<8;h20++){
                                    if(h20<3){
                                        hh[h20]=tt[h20];
                                    }else if(h20<5){
                                        hh[h20]=ff1+h20-3;
                                    }else{
                                        hh[h20]=tt[h20-2];
                                    }
                                }
                                for(h20=0;h20<8;h20++){
/*                                      printf("%d ",hh[h20]);*/
                                }
                            }else{
                                for(h21=0;h21<6;h21++){
                                    hh[h21]=tt[h21];
                                }
                            }
                        }else{
                            for(h22=0;h22<8;h22++){
                                hh[h22]=(h22+1)*10;
```

```
                        }

                }
        }else if(aa1==4){
                for(h23=0;h23<8;h23++){
                        hh[h23]=tt[h23];
                }

        }else{
                for(h24=0;h24<8;h24++){
                        hh[h24]=(h24+1)*20;
                }

        }

        p=hh;
        return p;
}
```

Hamming_LPC.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "svd.h"
#include "hamming_LPC.h"

double* hamming(double y[],int n1){


        int nn,c=110;
        int n2=(n1/c)-2;
        if(n2<=0){
                n2=1;
        }
        if(n2<=11){
                nn=11;
        }else{
                nn=n2;
        }
/*      printf("%d ",n2);*/
        int n=330,p1=10;
        double p[338][10],d[10],q[10][10];
        double pp1[nn][11],dd[11],qq[11][11];
        double *pp;
/*      double y[2200];*/
```

```
double h[330];
double h1[n2][330];
double y2[338];
double lpc[n2][11];
double R[338][10];
double R2[10][338];
double w1[10][338];
int m=338,n3=10;
int ww,ww1;
int tt,tt1,tt3,tt4,tt6,tt7,tt8,tt10,tt11;
double a1,a3,a4,toll;
int i,i1,i2,i3,j1,k,d1;


FILE *f1=fopen("/home/pi/MiProye/hamming.txt","r");
for(ww=0;ww<330;ww++){
        fscanf(f1,"%lf ",&h[ww]);
}
fclose(f1);



d1=0.0;
for (i=0;i<n2;i++){
        d1=c*i;
        for (k=0;k<n;k++){
                h1[i][k]=0;
                h1[i][k]=y[k+d1]*h[k];
        }
}

/*
**********************************************************************************************
******** */
/*      convolucion, recorte de la primera fila de la matriz de convolucion  y multiplicada por menos
uno*/
/*      matriz de 338 filas por 10 columnas*/
for(i1=0;i1<n2;i1++){
        for(i2=0;i2<p1;i2++){
                for(j1=0;j1<n+p1-2;j1++){
                        if  (j1<i2-1){
                                R[j1][i2]=0;
                        }else if(i2==0 && j1<n-1){
                                R[j1][i2]=h1[i1][j1+1]*(-1);
                        }else if (j1<n+i2-1 && j1>=i2-1){
                                R[j1][i2]=h1[i1][j1-i2+1]*(-1);
                        }else{
```

```
                                        R[j1][i2]=0;
                                }
                        }
                }

                for(ww1=0;ww1<333;ww1++){
                        y2[ww1]=R[ww1+1][0];
                }

                svd(338,10,R,p,d,q);
/*
/* ********************************************************************************************** */
                a1=0;
                for(i3=0;i3<n3;i3++){
                        if(d[i3]>a1){
                                a1=d[i3];
                        }
                }
                toll=0;
                if(a1<2){
                        toll=338*2.2204460492503/10000000000000000;
                }else if(a1<4){
                        toll=338*4.4408920985006/10000000000000000;
                }else if(a1<8){
                        toll=338*8.8817841970013/10000000000000000;
                }else if(a1<16){
                        toll=338*17.7635683940025/10000000000000000;
                }else{
                        toll=338*35.5271367880050/10000000000000000;
                }
                for(tt=0;tt<n3;tt++){
                        if(d[tt]<=toll){
                                break;
                        }
                }
                for(tt1=0;tt1<tt;tt1++){
                        d[tt1]=1/d[tt1];
                }
                for(tt3=0;tt3<tt;tt3++){
                        for(tt4=0;tt4<m;tt4++){
                                R2[tt3][tt4]=0;
                                R2[tt3][tt4]=d[tt3]*p[tt4][tt3];
                        }
                }
                for(tt6=0;tt6<tt;tt6++){
                        for(tt7=0;tt7<m;tt7++){
                                a3=0;
```

```
                                for(tt8=0;tt8<tt;tt8++){
                                        a3+=q[tt6][tt8]*R2[tt8][tt7];
                                }
                                w1[tt6][tt7]=0;
                                w1[tt6][tt7]=a3;
                        }
                }
                for(tt10=0;tt10<n3+1;tt10++){
                        a4=0;
                        if(tt10==0){
                                a4=1;
                        }else{
                                for(tt11=0;tt11<m-9;tt11++){
                                        a4+=w1[tt10-1][tt11]*y2[tt11];
                                }
                        }
                        lpc[i1][tt10]=0;
                        lpc[i1][tt10]=a4;

                }

        }/*proceso por cada fila de la matriz hamming */
        svd(n2,11,lpc,pp1,dd,qq);

        pp=dd;
        return pp;
}
```

svd.c

/*
Copyright (c) 2003, Division of Imaging Science and Biomedical Engineering,
University of Manchester, UK. All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list
of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or other
materials provided with the distribution.

Neither the name of the University of Manchester nor the names of its contributors
may be used to endorse or promote products derived from this software without

specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

```
#include <math.h>
#include <stdlib.h>
#include "svd.h"

//svd function
#define SIGN(u, v)     ( (v)>=0.0 ? fabs(u) : -fabs(u) )
#define MAX(x, y)     ( (x) >= (y) ? (x) : (y) )

static double   radius(double u, double v)
{
            double w;
            u = fabs(u);
            v = fabs(v);
            if (u > v) {
                        w = v / u;
                        return (u * sqrt(1. + w * w));
   } else {
      if (v) {
                  w = u / v;
          return (v * sqrt(1. + w * w));
                        } else
                                    return 0.0;
            }
}

/*
```

```
 Given matrix a[m][n], m>=n, using svd decomposition a = p d q' to get
 p[m][n], diag d[n] and q[n][n].
*/
void svd(int m, int n, double a[m][n], double p[m][n], double d[n], double q[n][n])

{
      int           flag, i, its, j, jj, k, l, nm, nm1 = n - 1, mm1 = m - 1;
      double        c, f, h, s, x, y, z;
      double        anorm = 0, g = 0, scale = 0;
      //double       *r = tvector_alloc(0, n, double);
                    double                        *r = (double*)malloc(sizeof(double)*n);

      for (i = 0; i < m; i++)
            for (j = 0; j < n; j++)
                  p[i][j] = a[i][j];
      //for (i = m; i < n; i++)
      //            p[i][j] = 0;

      /* Householder reduction to bidigonal form */
      for (i = 0; i < n; i++)
      {
            l = i + 1;
            r[i] = scale * g;
            g = s = scale = 0.0;
            if (i < m)
            {
                  for (k = i; k < m; k++)
                        scale += fabs(p[k][i]);
                  if (scale)
                  {
                        for (k = i; k < m; k++)
                        {
                              p[k][i] /= scale;
                              s += p[k][i] * p[k][i];
                        }
                        f = p[i][i];
                        g = -SIGN(sqrt(s), f);
                        h = f * g - s;
                        p[i][i] = f - g;
                        if (i != nm1)
                        {
                              for (j = l; j < n; j++)
                              {
                                    for (s = 0.0, k = i; k < m; k++)
                                          s += p[k][i] * p[k][j];
                                    f = s / h;
                                    for (k = i; k < m; k++)
```

```
                                p[k][j] += f * p[k][i];
                        }
                }
                for (k = i; k < m; k++)
                        p[k][i] *= scale;
        }
    }
    d[i] = scale * g;
    g = s = scale = 0.0;
    if (i < m && i != nm1)
    {
        for (k = l; k < n; k++)
            scale += fabs(p[i][k]);
        if (scale)
        {
            for (k = l; k < n; k++)
            {
                p[i][k] /= scale;
                s += p[i][k] * p[i][k];
            }
            f = p[i][l];
            g = -SIGN(sqrt(s), f);
            h = f * g - s;
            p[i][l] = f - g;
            for (k = l; k < n; k++)
                r[k] = p[i][k] / h;
            if (i != mm1)
            {
                for (j = l; j < m; j++)
                {
                    for (s = 0.0, k = l; k < n; k++)
                        s += p[j][k] * p[i][k];
                    for (k = l; k < n; k++)
                        p[j][k] += s * r[k];
                }
            }
            for (k = l; k < n; k++)
                p[i][k] *= scale;
        }
    }
    anorm = MAX(anorm, fabs(d[i]) + fabs(r[i]));
}

/* Accumulation of right-hand transformations */
for (i = n - 1; i >= 0; i--)
{
    if (i < nm1)
```

```
        {
            if (g)
            {
                for (j = l; j < n; j++)
                    q[j][i] = (p[i][j] / p[i][l]) / g;
                for (j = l; j < n; j++)
                {
                    for (s = 0.0, k = l; k < n; k++)
                        s += p[i][k] * q[k][j];
                    for (k = l; k < n; k++)
                        q[k][j] += s * q[k][i];
                }
            }
            for (j = l; j < n; j++)
                q[i][j] = q[j][i] = 0.0;
        }
        q[i][i] = 1.0;
        g = r[i];
        l = i;
    }
    /* Accumulation of left-hand transformations */
    for (i = n - 1; i >= 0; i--)
    {
        l = i + 1;
        g = d[i];
        if (i < nm1)
            for (j = l; j < n; j++)
                p[i][j] = 0.0;
        if (g)
        {
            g = 1.0 / g;
            if (i != nm1)
            {
                for (j = l; j < n; j++)
                {
                    for (s = 0.0, k = l; k < m; k++)
                        s += p[k][i] * p[k][j];
                    f = (s / p[i][i]) * g;
                    for (k = i; k < m; k++)
                        p[k][j] += f * p[k][i];
                }
            }
            for (j = i; j < m; j++)
                p[j][i] *= g;
        } else
            for (j = i; j < m; j++)
                p[j][i] = 0.0;
```

```
        ++p[i][i];
    }
    /* diagonalization of the bidigonal form */
    for (k = n - 1; k >= 0; k--)
    {                   /* loop over singlar values */
        for (its = 0; its < 30; its++)
        {               /* loop over allowed iterations */
            flag = 1;
            for (l = k; l >= 0; l--)
            {       /* test for splitting */
                nm = l - 1;     /* note that r[l] is always
                                * zero */
                if (fabs(r[l]) + anorm == anorm)
                {
                    flag = 0;
                    break;
                }
                if (fabs(d[nm]) + anorm == anorm)
                    break;
            }
            if (flag)
            {
                c = 0.0;        /* cancellation of r[l], if
                                * l>1 */
                s = 1.0;
                for (i = l; i <= k; i++)
                {
                    f = s * r[i];
                    if (fabs(f) + anorm != anorm)
                    {
                        g = d[i];
                        h = radius(f, g);
                        d[i] = h;
                        h = 1.0 / h;
                        c = g * h;
                        s = (-f * h);
                        for (j = 0; j < m; j++)
                        {
                            y = p[j][nm];
                            z = p[j][i];
                            p[j][nm] = y * c + z * s;
                            p[j][i] = z * c - y * s;
                        }
                    }
                }
            }
            z = d[k];
```

```
if (l == k)
{       /* convergence */
        if (z < 0.0)
        {
                d[k] = -z;
                for (j = 0; j < n; j++)
                        q[j][k] = (-q[j][k]);
        }
        break;
}
if (its == 30)
{
        //error("svd: No convergence in 30 svd iterations", non_fatal);
        return;
}
x = d[l];       /* shift from bottom 2-by-2 minor */
nm = k - 1;
y = d[nm];
g = r[nm];
h = r[k];
f = ((y - z) * (y + z) + (g - h) * (g + h)) / (2.0 * h * y);
g = radius(f, 1.0);
/* next QR transformation */
f = ((x - z) * (x + z) + h * ((y / (f + SIGN(g, f))) - h)) / x;
c = s = 1.0;
for (j = l; j <= nm; j++)
{
        i = j + 1;
        g = r[i];
        y = d[i];
        h = s * g;
        g = c * g;
        z = radius(f, h);
        r[j] = z;
        c = f / z;
        s = h / z;
        f = x * c + g * s;
        g = g * c - x * s;
        h = y * s;
        y = y * c;
        for (jj = 0; jj < n; jj++)
        {
                x = q[jj][j];
                z = q[jj][i];
                q[jj][j] = x * c + z * s;
                q[jj][i] = z * c - x * s;
        }
```

```
                    z = radius(f, h);
                    d[j] = z;       /* rotation can be arbitrary
                                    * id z=0 */
                    if (z)
                    {
                        z = 1.0 / z;
                        c = f * z;
                        s = h * z;
                    }
                    f = (c * g) + (s * y);
                    x = (c * y) - (s * g);
                    for (jj = 0; jj < m; jj++)
                    {
                        y = p[jj][j];
                        z = p[jj][i];
                        p[jj][j] = y * c + z * s;
                        p[jj][i] = z * c - y * s;
                    }
                }
                r[l] = 0.0;
                r[k] = f;
                d[k] = x;
            }
        }
        free(r);

}
```

Interp1.c

```
#include <stdio.h>
#include <float.h>
#include <stdlib.h>
#include "interp1.h"

int findNearestNeighbourIndex( double value, double *x, int len )
{
        double dist;
        int idx;
        int i;
        idx = -1;
        dist = DBL_MAX;
        for ( i = 0; i < len; i++ ) {
                double newDist = value - x[i];
                if ( newDist >= 0 && newDist < dist ) {
                        dist = newDist;
```

```
                        idx = i;
                }
        }
        return idx;
}
void interp1(double x[44], int x_tam, double y[44], double xx[440], int xx_tam, double *yy)
{
        double dx, dy, *slope, *intercept;
        int i, indiceEnVector;
        slope=(double *)calloc(x_tam,sizeof(double));
        intercept=(double *)calloc(x_tam,sizeof(double));
        for(i = 0; i < x_tam; i++){
                if(i<x_tam-1){
                        dx = x[i + 1] - x[i];
                        dy = y[i + 1] - y[i];
                        slope[i] = dy / dx;
                        intercept[i] = y[i] - x[i] * slope[i];
                }else{
                        slope[i]=slope[i-1];
                        intercept[i]=intercept[i-1];
                }
        }
        for (i = 0; i < xx_tam; i++) {
                indiceEnVector = findNearestNeighbourIndex( xx[i], x, x_tam);
                if (indiceEnVector != -1)
                        yy[i]=slope[indiceEnVector] * xx[i] + intercept[indiceEnVector];
                else
                        yy[i]=DBL_MAX;
        }
        free(slope);
        free(intercept);
}
```

Ordenar.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "ordenar.h"

double* ordenar(double h[]){
        int i,j,a1;
        double *pp,a;
        static double h1[11];
        for(j=0;j<11;j++){
                a=a1=0;
```

```
                for(i=0;i<11;i++){
                        if(h[i]>a){
                                a=h[i];
                                a1=i;
                        }
                }
                h[a1]=0;
                h1[j]=a;
        }
        pp=h1;
        return pp;
}
```

Distancia.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "distancia.h"

double* distancia(double y[], double y1[][431]){
        int i,j;
        double a,*p;
        static double h[5];
        for(i=0;i<5;i++){
                a=0.0;
                for(j=0;j<431;j++){
                        a+=(y[j]-y1[i][j])*(y[j]-y1[i][j]);
                }
                h[i]=sqrt(a);
        }
        p=h;
        return p;
}
```

Módulo de reconocimiento de palabras basado en LPC para el control de desplazamiento de dispositivos con micromotores

Word recognition module based on LPC for controlling the movement of devices with micromotors

Luis Felipe González[1], Ana Yulieth Perdomo[2], y Martín Diomedes Bravo[3]

Resumen

Este documento presenta el diseño e implementación de un sistema de reconocimiento de palabras con el propósito de controlar el movimiento de un modelo escala de silla de ruedas, aplicando como método de referencia, los coeficientes de predicción lineal (LPC) para la extracción de las características de la señal de voz y a su vez el uso de la función de distancia euclidiana para la comparación de patrones. Para la implementación del módulo se utilizó una raspberry pi, en la cual se implementa el algoritmo de reconocimiento de los comandos y la ejecución del control de los servomotores.

Palabras clave: LPC, Patrones, Distancia Euclidiana, Raspberry pi, Servomotores.

Abstract

This document presents the design and implementation of a word recognition system with the purpose of controlling the movement of a wheelchair scale model, applying as a reference method, the linear prediction coefficients (LPC) for the extraction of the characteristics of the voice signal and in turn the use of the Euclidean distance function for the patterns comparison. For the implementation a module of raspberry pi was used, in which the algorithm of recognition of the commands and the execution of the control of the servomotors was implemented.

Keywords: LPC, Patterns Euclidean distance, Raspberry pi, Servomotors.

[1] *Estudiante de Ingenieria Electronica, Universidad Surcolombiana Neiva. Av. Pastrana Carrera 1. U20112104346@usco.edu.co*
[2] *Estudiante de Ingenieria Electronica, Universidad Surcolombiana Neiva. Av. Pastrana Carrera 1. U20112104546@usco.edu.co*
[3] *Magíster en Telecomunicaciones, Universidad Surcolombiana Neiva. Av. Pastrana Carrera 1. martin.bravo@usco.edu.co*

1. Introducción

Las nuevas tecnologías para el reconocimiento de voz están revolucionando la forma en que el usuario recibe y procesa la información automatizando sus diferentes actividades (Keimel, 2016), esto conlleva a una mejora en la calidad de vida si se piensa en las personas con alguna discapacidad. Además, mediante redes neuronales y usando software automatizados permiten a usuarios con dificultades de audición o ausencia total de ésta, la posibilidad de emplearlo en primera instancia como una herramienta inicial de apoyo para el aprendizaje de sílabas. (San Juan, *et al*.,2016) Cuando se habla de un control con voz lo primero que se considera es el reconocimiento, es decir, que el sistema entienda las palabras mas no a su significado y con base en este ejecutar una acción.

Crear un control con voz es una tarea desafiante y muy importante, se basa principalmente en la precisión, la API, el rendimiento, la velocidad en tiempo real, el tiempo de respuesta y la compatibilidad., por lo tanto, consideramos los sistemas de reconocimientos de voz dependiendo de las necesidades del usuario (Matarneh, *et al.*, 2017).

Entre los trabajos realizados en esta área es de resaltar un trabajo de reconocimiento de voz humana mediante el uso de DSP con codificación lineal predictiva (LPC). El sistema domótico controla diversos elementos cotidianos ubicados en el hogar a través del protocolo de comunicaciones IEEE 802.15.4 con radios zigbeePro, realizado en la universidad de San Buenaventura (Camargo, *et al.*, 2008), otra aplicación de voz con LPC fue implemetada en la universidad de pinar del rio de Cuba con un diseño de un codec de audio para telefonía VoIP implementado sobre un microprocesador MicroBlaze empotrado sobre un circuito FPGA de la familia Spartan 3E. (García, 2011), y por último otro trabajo de sistemas de reconocimiento basados en LPC se encuentra en la universidad Pedagogica y tecnologica de Colombia en donde se utilizo una FPGA que permite activar la ejecución de al menos tres tareas diferentes para el robot ‘Rover 5’ del grupo GIRA de la UPTC. (Bustos & Pantevis, 2016).

En este trabajo de reconocimiento de palabras con codificación lineal predictiva (LPC) se utilizó una raspberry pi 3 como medio de procesamiento del algoritmo de reconocimiento y el control de movimiento de la silla de ruedas a escala 1:3, donde los comandos utilizados son izquierda, derecha, atrás, adelante y alto. También dentro de los procesos matemáticos realizados cabe resaltar la descomposición en valores singulares (SVD) que permite simplificar el proceso de comparación de patrones.

2. Materiales y metodos

2.1 Procesamiento de la señal de voz

Los sistemas de reconocimiento de voz comprenden diferentes disciplinas y tienen en común la etapa inicial conocida como el procesamiento de señales, el cual convierte la señal de voz en alguna representación paramétrica para su posterior análisis.

El sistema de reconocimiento de palabras implementado tiene como base los siguientes procesos para la realización del algoritmo.

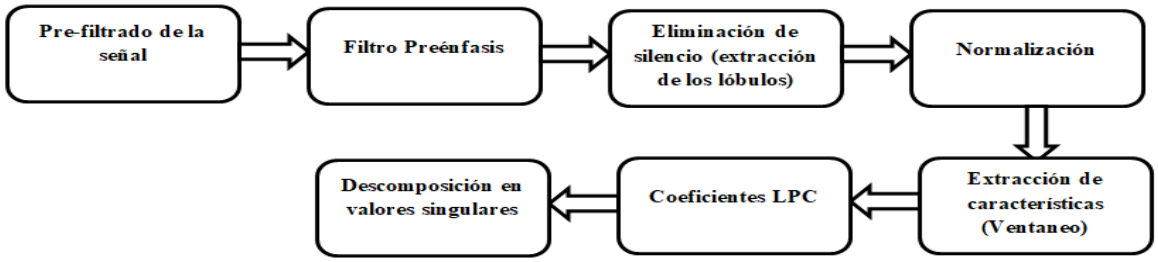


Figura 1 Procesamiento de la señal de voz

En la figura 2 se muestra la etapa de entrenamiento que realiza la extracción de patrones de cada uno de los comandos del banco de palabras que son los comandos de voz tomados de un grupo de personas, los cuales son procesados y guardados en un banco de datos para su posterior comparación.

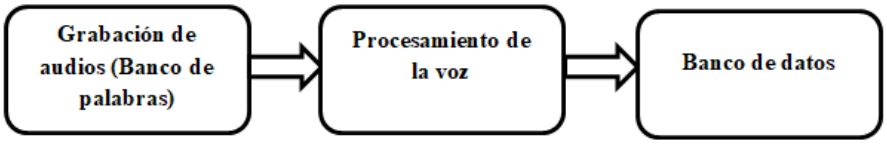


Figura 2 Etapa de entrenamiento

En la figura 3 se muestra la etapa de reconocimiento que realiza la extracción de características del comando dicho por el usuario y lo compara con el banco de palabras por medio de la distancia euclidiana.

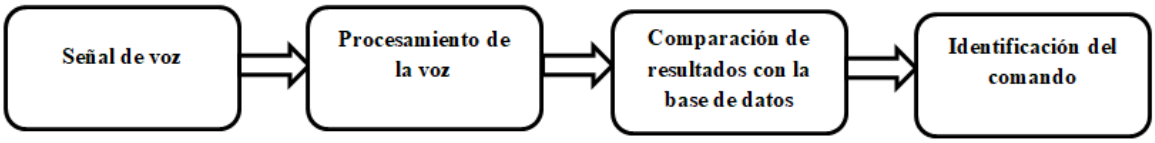


Figura 3 Etapa de reconocimineto

2.1.1 Obtención del banco de palabras

El sistema de reconocimiento tiene cinco comandos para su funcionamiento (derecha, izquierda, atrás, alto y adelante), la señal de voz se muestrea a un rango de frecuencias entre 8 y 16 KHz. Para el desarrollo del algoritmo se recogieron grabaciones con un tiempo de 2 segundos y con una frecuencia 11025 Hz la cual está por encima del doble de la frecuencia máxima de la voz humana, se utilizó un formato WAV, y se tomaron 80 muestras de audios de hombres y mujeres de diferentes edades.

2.1.2 Pre-filtrado de la señal

En sistemas de comunicación, los filtros se utilizan para sintonizar frecuencias específicas y eliminar otros. En los sistemas de procesamiento de señales digitales usan filtros para evitar aliasing de ruido e interferencia fuera de banda (Moreno, 2004).

El filtro butterworth es un tipo de diseño de filtro de procesamiento de señal y está diseñado para tener una respuesta de frecuencia lo más plana matemáticamente posible en la banda de paso. De esta manera la realización de un pre-filtrado a las señales de voz utilizando este tipo de filtro pasa banda de orden 4 con frecuencias de corte 40 Hz y 1000 Hz es necesario con el fin de eliminar ciertas perturbaciones y soportar el proceso de reconocimiento.

2.1.3 Filtro preénfasis

Un filtro preénfasis es un método de procesamiento de señales diseñado para aumentar la magnitud de algunas frecuencias con respecto a las de otras con el fin de mejorar la relación señal a ruido, disminuyendo así los efectos como la atenuación y distorsion de la señal, este filtro esta representado por la ecuación 1.

$$H(z) = (1 - az^{-1})$$

Ecuación 1 Filtro preénfasis

Donde a es igual 0.95, la aplicación de este filtro asegura al sistema espectralmente aplane la señal y no se pierda información (Rueda, 2011).

2.1.4 Eliminación de silencio (Extracción de lóbulos)

La palabra lóbulo en este trabajo hace referencia a un tramo de la señal de voz que identifica un fonema. Los segmentos de silencio de la señal digital son removidos tomando como base un valor umbral, extrayendo solo las partes con mayor energía para su posterior análisis. Para el cálculo de la energía se aplica la ecuación 2.

$$E = \frac{1}{N}\sum_{k=1}^{N} x[k]^2$$

Ecuación 2 Calculo de la energía

Donde $x[k]$ es la señal de voz, N es el número de muestras y E es la energia de la señal (Corintios, 2009).

2.1.5 Normalización

La normalización es una operación estadística y se usa para escalar valores heterogéneo, en este caso aplicarlo facilita la definición de umbrales en diferentes algoritmos. Finalmente, el rango de datos disminuye y se limita desde -1 hasta 1. La ecuación que define este proceso está dada por la ecuación 3.

$$Norma = x/\max(x)$$

Ecuación 3 Normalizacion

Donde x representa cada valor de la señal.

2.1.6 Ventaneo

En el sistema de reconocimiento de palabras se cuenta con grabaciones de diferentes personas sometidas a un pre-procesado adecuado, para así proceder a la extracción de características de la señal de voz de acuerdo a ciertos parámetros.

Se entiende que la señal de voz contiene numerosas variaciones y debido a ello la extracción de características se debe realizar en intervalos entre 20 ms y 30 ms (San Martin & Carrillo, 2004), para esto es oportuno aplicar una ventana a la señal de un tamaño conveniente de acuerdo a la frecuencia de muestreo, en este caso se utiliza una frecuencia de muestreo de 11025Hz a intervalos de 30 ms lo que equivale a 330 muestras por ventana con un incremento de 110 muestras aproximadamente, la ecuacion 4 muestra la representacion general de esta ventana.

$$V(n) = \begin{cases} 0.54 - 0.46\, x \cos\left(\frac{2\pi n}{N}\right) & si\ 0 \le n \le N \\ 0 & En\ otro\ caso \end{cases}$$

Ecuación 4 Ventana Hamming

Donde n representa los valores de la señal y N el número de muestras.

2.1.7 Coeficientes LPC

Al emplear el ventaneo posteriormente se procede a aplicar las técnicas de extracción de patrones que son los coeficientes LPC (codificación por predicción lineal), sus parámetros se ajustan a las características del tracto vocal y representa la envolvente espectral de la señal de forma comprimida (Baquero, *et al.*, 2011). Partiendo de la idea que se puede predecir la muestra presente a partir de una combinación lineal de las muestras pasadas, generando una descripción espectral basada en segmentos cortos de señal considerando una salida s[n] a una respuesta de un filtro todo-polos que tiene como entrada u[n]. La función de transferencia del filtro se describe en la ecuacion 5.

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{G}{A(z)}$$

Ecuación 5 Funcion de transferencia del filtro LPC

Donde G es el parámetro de ganancia, a_k los coeficientes del filtro y p el orden de dicho filtro. La señal filtrada s[n] se relaciona mediante la excitación de la entrada u[n].

El error de prediccion esta dado por la ecuacion 6.

$$[z] = y[z](1 - \sum_{k=1}^{p} a_k z^{-k})$$

Ecuación 6 Error de predicción

2.1.8 Descomposición de valores singulares

Obteniendo los coeficientes LPC de cada lóbulo se procede a la etapa de reconocimiento de las palabras establecidas calculando los valores singulares (SVD), La teoría muestra que si tenemos una matiz A mxn, los valores singulares de A son las raices cuadradas de los autovalores de $A^T A$, y se denotan mediante $\sigma_1, \dots, \sigma_n$. Es una convención acomodar los valores singulares de modo que $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. Se puede demostrar matemáticamente que una matriz A de mxn ($m \geq n$) se puede factorizar como se muestra en la ecuación 7.

$$A = U \sum V^T$$

Ecuación 7 Factorizacion de la matriz A.

Donde U es una matriz con columnas ortogonales de mxn, V es una matriz ortogonal de nxn, y Σ una matriz diagonal de nxn. (Demmel, 1997).

2.1.9 Reconocimiento (Comparación de los resultados)

Para que la comparación de resultados sea posible es necesario que los vectores de patrones tengan la misma longitud, debido a que el resultado del ventaneo de cada lóbulos entrega como resultado una matriz de 10 columnas que corresponde al valor del LPC y un numero indefinido de filas que depende de la longitud del lóbulo, por este motivo se aplica la diagonal de valores singulares del SVD la cual simplifica dicha matriz en un vector de 10 patrones por lóbulo, estos 10 valores son puestos en una matriz de ceros con una longitud de 40 que corresponde al número máximo de lóbulos, esta ultima matriz es interpolada para tener más valores al mometo de compararlos, este proceso se le realiza al comando dicho por el usuario y a los patrones de referencia previamente almacenados

(base de datos) en memoria usando una medida de similitud, la distancia euclidiana permite hallarlo mediante la ecuación 8.

$$d = \sqrt{\sum_{i=1}^{n}(pi - qi)^2}$$

Ecuación 8. Distancia Euclidiana

Donde d es la distancia, pi es los valores finales de cada comando y qi los valos del usuario obtenido en el momento de dar la acción (Bedolla, 2011).

2.2. Software

El algoritmo se implemento utilizando dos lenguajes de programación: Python y C, los cuales trabajan de forma alterna. El primero maneja el programa principal el cual se encarga de tomar la señal de voz, llamar el ejecutable que contiene el proceso de reconocimiento en lenguaje C y el control de los micromotores. El segundo realiza el proceso de reconocimiento del comando. El programa principal recoge la señal de voz en un periodo de tiempo de dos segundos a una frecuencia de 11025 Hz. El audio es almacenado en formato wave para luego ser convertido en un vector y guardado en archivo formato de texto, esta conversión se hace para que el programa de reconocimiento implementado en lenguaje C lo pueda leer, procesar y al final entregar un formato igual donde se tiene la información que corresponde al comando reconocido y así después ser interpretado por Python y realizar la función de control, este proceso se muestra en la figura 4.

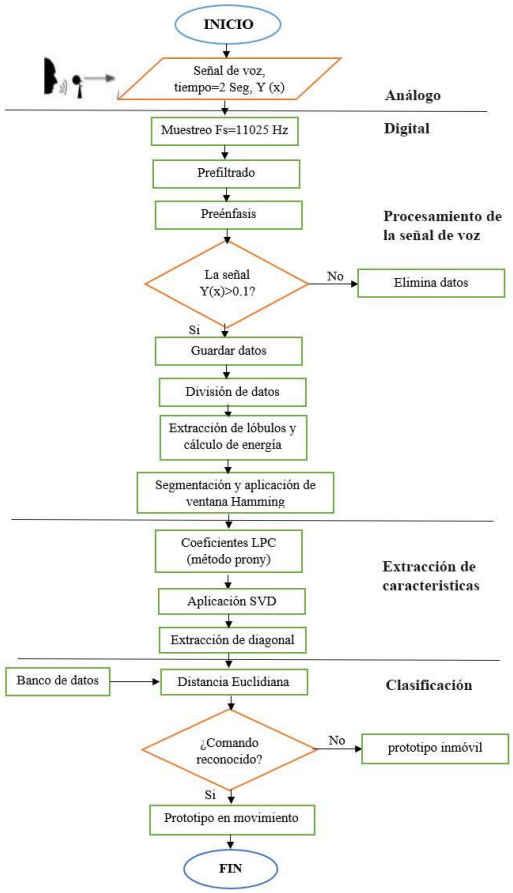


Figura 4 Diagrama de flujo para el reconocimiento de palabras.

2.2.3 Control de los servomotores

El control se realiza desde el programa principal que está desarrollado en Python. El proceso comienza leyendo la información guardada en el formato texto, la cual indica que acción realizar, la información que se encuentra allí es un número del cero al cinco, donde cero indica que no se reconoció el comando, y de uno a cinco corresponde a las acciones de movimiento como se muestra en la tabla 1.

Tabla 1. Comandos

Numero	Comando
0	No reconocido
1	Izquierda
2	Derecha
3	Atrás
4	Adelante
5	Alto

Cada acción está asociada a cuatro pines de la Raspberry pi excepto cuando no se reconoce el comando, dos corresponden al motor derecho y los otros dos al motor izquierdo.

La lógica se basa en poner en alto o bajo algunas salida de la raspberry pi dependiendo del comando requerido por el usuario y haciendo que el prototipo realice la acción pedida. Los pines 11 y 12 son los encargados de controlar el movimiento del motor izquierdo, y los pines 13 y 15 el motor derecho.

El funcionamiento del prototipo esta hecho de la siguiente manera:

- Si la orden requerida es adelante, el prototipo avanzara de forma permanente hasta que una nueva orden sea reconocida.
- Si la orden requerida es atrás, el prototipo retrocederá de forma permanente hasta que una nueva orden sea reconocida.
- Si la orden requerida es derecha y el prototipo esta en movimiento el motor derecho se detendrá por un segundo cambiando de dirección, pero si está detenido se accionará el motor izquierdo por el mismo tiempo.
- Si la orden requerida es izquierda y el prototipo esta en movimiento el motor izquierdo se detendrá por un segundo cambiando de dirección, pero si está detenido se accionará el motor derecho por el mismo tiempo
- Si la orden requerida es alta y el prototipo esta en movimiento se detendrá.

2.4. Hardware

El diseño del módulo de reconocimiento de palabras es mostrado en la figura 5, donde se puede observar los componentes descritos anteriormente.

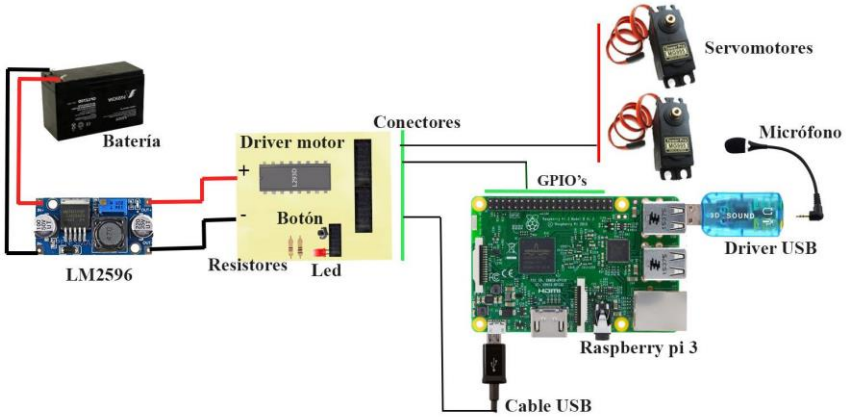


Figura 5 Sistema del reconocimiento de palabras

Para la protección del sistema se diseña una caja en acrílico con dimensiones de base de 10 x 12 cm y una altura de 5cm como se muestra en la figura 6.

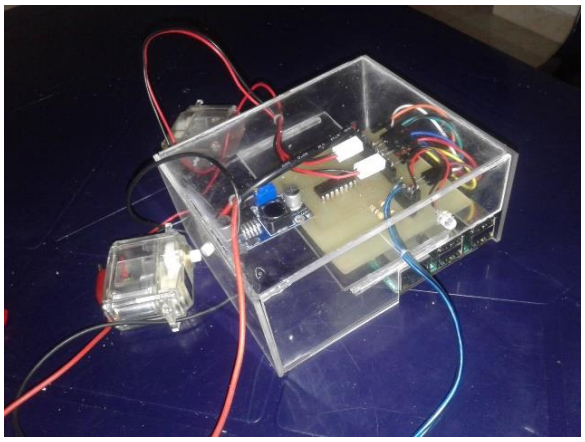


Figura 6 Módulo de reconocimiento de palabras

Para la realización de las pruebas se utiliza un prototipo de silla de ruedas 1:3 adquirida de la universidad Sur colombiana. En la figura 7 se muestra la implementación.

Figura 7 Implementación del módulo de reconocimiento de palabras en una silla de ruedas

3. Resultados y discusión

3.1 Recomendaciones para el manejo del prototipo

A continuación, se exponen los pasos que se deben llevar a cabo para la manipulación del prototipo de silla de ruedas.

- ✓ Verificar que la batería esté conectada al prototipo de silla de ruedas.
- ✓ Activar el interruptor de encendido y esperar 20 segundos para que se inicie el sistema.
- ✓ Presionar el botón que se encuentra al lado del micrófono e inmediatamente se encenderá un led que indica que puede dar el comando de la acción que quiere que se realice.
- ✓ Al momento de pronunciar el comando de voz procure hacerlo de forma pausada y clara, para así tener una mejor respuesta del sistema.
- ✓ El led que indica el momento de pronunciar el comando tiene un tiempo de encendido de dos segundos y cuando se apague se realiza la acción.
- ✓ Si el sistema no responde o responde incorrectamente espere a que el led se apague para presionar el botón y decir el comando de voz nuevamente.

3.2 Pruebas de funcionamiento

En la realización del banco de datos fue necesaria la recolección de audios de 10 hombres y 10 mujeres de diferentes edades y tono de voz, donde se tomó un audio por comando.

Para analizar el rendimiento del sistema se realizaron pruebas en dos ambientes diferentes, uno en un lugar silencioso y otro con ruido. La tabla 2 y tabla 3 muestran los resultados obtenidos al pronunciar 10 veces cada palabra por hombres y mujeres

Tabla 2. Resultados obtenidos en un ambiente silencioso

Comando	Hombre		Mujer	
	Acertada	Error	Acertada	Error
Adelante	9	1	8	2
Derecha	8	2	8	2
Izquierda	9	1	8	2
Atrás	8	2	9	1
Alto	9	1	9	1

Tabla 3. Resultados obtenidos en un ambiente ruidoso

Comando	Hombre		Mujer	
	Acertada	Error	Acertada	Error
Adelante	8	2	7	3
Derecha	7	3	8	2
Izquierda	7	3	7	3
Atrás	8	2	7	3
Alto	8	2	8	2

Las siguientes graficas muestran el nivel porcentual de los comandos reconocidos en las pruebas realizadas en los dos ambientes con la participación de hombres y mujeres.

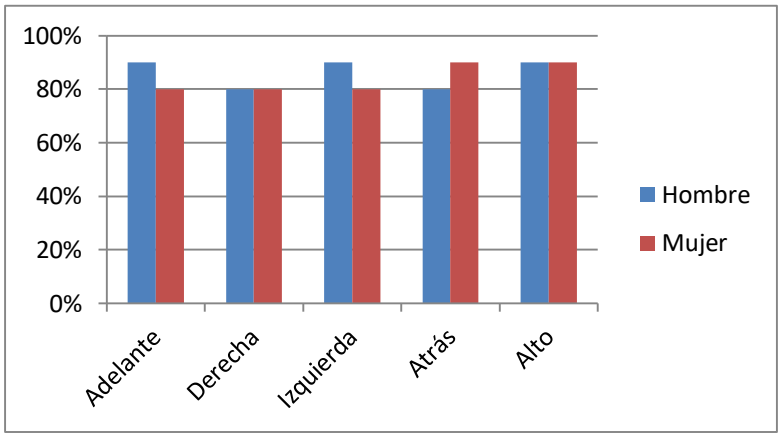


Figura 8 Resultados porcentuales obtenidos en un ambiente en silencio

En la figura 8 se muestra los resultados obtenidos en un ambiente silencioso, donde se tiene que el mayor porcentaje es de un 90% y el menor de un 80% y con un promedio en los hombres de un 86% y el en las mujeres un 84% en el que se obtiene un promedio general de un 85%. Estos resultados indican que el sistema es fiable y bueno.

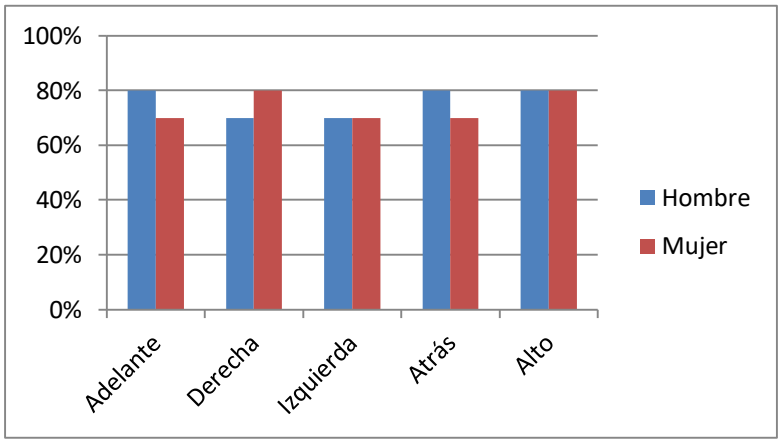


Figura 9 Resultados porcentuales obtenidos en un ambiente con ruido

En la figura 9 se muestra los resultados obtenidos en un ambiente con ruido, donde se tiene que el mayor porcentaje es de un 80% y el menor de un 70% y con un promedio en los hombres de un 76% y el en las mujeres un 74% en el que se obtiene un promedio general de un 75%. Estos resultados indican que el sistema sigue siendo fiable.

3.3 Métodos utilizados para el mejoramiento del reconocimiento de palabras

Para el mejoramiento del sistema se hicieron dos procesos adicionales a la extracción de los coeficientes de predicción lineal LPC que es la matriz diagonal de valores singulares del SVD y la interpolación de estos valores.

En el proceso de extracción de características se realiza un análisis por cada lóbulo de la señal de voz, donde se obtuvo como resultado una matriz de coeficientes LPC por cada uno. Esta matriz contiene un número definido de columnas el cual hace referencia a los 10 coeficientes LPC por ventana Hamming que se aplicó y un número indeterminado de filas que depende de la cantidad de desplazamientos de la ventana hasta completar la longitud del lóbulo, esto hace que sea imposible calcular la distancia euclidiana si se tiene en cuenta que los vectores tienen que ser del mismo tamaño. Por esta razón se calcula la matriz diagonal de valores singulares la cual me entrega un vector de diez muestras por cada matriz de LPC, posteriormente se calcula una interpolación lineal para tener una mejor definición de las curvas formadas por los promedios de las diagonales de los valores singulares de cada comando de voz.

4. Conclusiones

El modelo LPC es un método que permite obtener la información más importante de un contenido extenso de muestras de la señal de voz, de tal manera que el proceso de reconocimiento se desarrolle en menor.

El desarrollo de un algoritmo de reconocimiento de palabras es una tarea desafiante debido a los diferentes parámetros que se deben tener en cuenta y el procesamiento que se implementa a la señal de voz, para ello se tiene que utilizar técnicas y conceptos que permitan alcanzar una mayor velocidad computacional con un menor gasto de memoria, puesto que esto influye en el rendimiento del sistema.

Para que el sistema de reconocimiento sea robusto fue necesario tomar diferentes muestras de audios de distintas personas para obtener un rango de eficiencia más alto.

El módulo de reconocimiento de palabras se implementó en el prototipo de silla de ruedas cumpliendo con el objetivo expuesto, además puede ser implementado en otros sistemas donde se requieran estos comandos, también tiene la versatilidad de aumentar o disminuir el número de comandos dependiendo de las acciones que se quieran realizar.

La computadora de placa simple Raspberry Pi 3 es una buena opción para el desarrollo e implementación del algoritmo del sistema de reconocimiento de palabras, debido a que permite un funcionamiento eficiente en los dos lenguajes de programación utilizados a un tiempo de ejecución rápido.

5. Referencias bibliográficas

Keimel, M., 2016, Integración de dispositivos en una plataforma de vida cotidiana asistida por computadora, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina, URI: http://www.ridaa.unicen.edu.ar/xmlui/handle/123456789/646.

San Juan, E. Jamett, M. Kaschel, H. Sánchez, L.2016, Sistema de reconocimiento de voz mediante wavelets, predicción lineal y redes backpropagation, Departamento de Ingeniería Eléctrica. Universidad de Santiago de Chile, Chile. DOI: http://dx.doi.org/10.4067/S0718-33052016000100002

Matarneh, R. Lyashenko, V. Maksymova, S. Belova, N. 2017, Speech Recognition Systems: A Comparative Review , Universidad de Radioelectrónica, Kharkiv, Ucrania. DOI: 10.9790/0661-1905047179.

Camargo, J. García, L. Gaona, E., 2012, Reconocimiento de voz humana aplicado a la domótica, Universidad de San Buenaventura, Colombia, DOI: http://dx.doi.org/10.21500/01247492.1285

García, A., Gallego, E., Domínguez, M., Correa, A., Rodríguez, J., 2011, Codificación de voz mediante coeficientes de predicción lineal (LPC) sobre Microblaze, Universidad de Pinar del Rio, Cuba.

Bustos, F., Pantevis, H., 2016, Implementación de un sistema de reconocimiento de voz en FPGA como interfaz Hombre-Máquina en aplicaciones, Universidad Pedagógica y tecnológica de Colombia, Colombia, URI: https://repositorio.uptc.edu.co/handle/001/1869

Moreno, I., 2004, Apuntes de instrumentación electrónica, Área de tecnología Electrónica Universidad de Burgos, España. Chapter 6.

Rueda, L., 2011, Mejoras en reconocimiento del habla basadas en mejoras en la parametrización de la voz, Universidad Autónoma de Madrid, España. URI: hdl.handle.net/10486/6734.

Corintios, M., 2009, Signals, Systems, Transforms, and Digital Signal Processing with MATLAB. Chapter 12: Energy and Power Spectral Densities. Taylor & Francis Group, ISBN-13: 978-1420090482.

San Martin, C., Carrillo, R., 2004, Implementación de un reconocedor de palabras aisladas independientes del locutor, Chile, pp. 9-14. DOI: dx.doi.org/10.4067/S0718-13372004000100002.

Baquero, Y., Alezones, Z., Borrero, H., 2011, Robot móvil controlado por comandos de voz LPC DTW, Universidad de los Llanos, Villavicencio, Colombia, pp.15-25. DOI: 10.14483/22484728.3524.

Demmel, J., 1997, Applied Numerical Linear Algebra, SIAM, Philadelphia, ISBN: 9780898713893.

Bedolla, J., 2011, Aplicación de distancias entre términos para datos planos y jerárquicos, Universidad Politécnica de Valencia, Valencia, España. URI: hdl.handle.net/10251/15874.