ESCUELA POLITÉCNICA DEL EJÉRCITO EXTENSIÓN LATACUNGA

INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

DISPOSITIVO DE RECONOCIMIENTO DE VOZ EN PERSONAS CON DISCAPACIDAD EN EL HABLA

MARIELA ALEXANDRA ALENCASTRO VACA

TANIA DEL ROCÍO DE LA CRUZ TAPIA

TESIS PRESENTADA COMO REQUISITO PREVIO A LA OBTENCIÓN DEL GRADO DE

INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

AÑO 2011

CERTIFICACIÓN

Tras examinar las pruebas del dispositivo de reconocimiento de voz para personas con discapacidad en el habla, se certifica que el presente trabajo fue desarrollado en su totalidad por la Srta. Alencastro Vaca Mariela Alexandra junto a la Srta. De La Cruz Tapia Tania del Rocío, bajo nuestra supervisión.

_____

Ingeniera Nancy Guerron (DIRECTORA DEL PROYECTO)

_____

Ingeniero Armando Álvarez (CODIRECTOR DEL PROYECTO)

AGRADECIMIENTO

La presente Tesis está dedicada a mis padres porque gracias al apoyo incondicional y amor que ellos me han brindado, he podido alcanzar las metas que me he propuesto.

A mis hermanos por todo el apoyo y amor que me han brindado.

A mis amigos Mary, Alexandra, Jorge, Neto, Lili por estar siempre a mi lado cuando los he necesitado.

A mis abuelitos ya que ellos me han brindado su apoyo incondicional, me dieron consejos para poder alcanzar todo lo que me propuesto, por haberme enseñado lo importante de la vida.

A Dios por haberme dado la oportunidad de vivir y conocer a gente tan maravillosa como la que he conocido hasta el día de hoy.

Tania.

AGRADECIMIENTO

En primer lugar agradezco a Dios, a La Virgen María, y al Ángel de la guarda por haberme permitido la gracia de haber nacido en una familia que siempre me ha llenado de su amor y cariño, por haberme llevado por los caminos que si no había entendido antes hoy veo que no se equivocaron y siempre han anticipado cada paso, cada pensamiento, cada sentimiento, que inspirados en ellos espero seguir creciendo más en la fe y amor para mejorar como persona y profesional.

A mis padres por ser ellos quienes me han apoyado cuando ya no creía poder seguir, porque con sus acciones lograron inspirar en mí ser una mejor persona cada día, por su amor y confianza que me ayudaron a llegar a cumplir mis objetivos.

A mis hermanos, que a pesar de ser menores siempre me enseñaban cosas nuevas con su cariño y amor, que se mantiene a mi lado pese a las adversidades.

A mis amigos, infinitas gracias les doy por ayudarme cuando más los necesitaba, por ser quienes estaban presente en tristezas, alegrías y desvelos. Por todo lo vivido en el transcurso de nuestra preparación académica, donde comprendí que no solo se encontraba la sabiduría si no también gran tesoro que es su amistad.

Por último y no menos importante a mis queridos catedráticos, con sus enseñanzas fueron fomentando el conocimiento en cada alumno que llego a este noble establecimiento, que supieron inspiran la investigación y el anhelo de aprendizaje.

A cada uno de las personas que marcaron una huella en mi vida, gracias porque dejaron una enseñanza.

Gracias!

Alexandra.

ÍNDICE DE CONTENIDOS

RESUMEN....1

CAPITULO I....2

FUNDAMENTACIÓN TEÓRICA....2

1.1 INTRODUCCION....2

1.2 FUNDAMENTOS DEL HABLA....2

1.3 FUNDAMENTOS DE LA ADQUISICIÓN DE DATOS....13

1.4 SOFTWARE MATLAB....25

1. 5 PROCESAMIENTO DE LA SEÑAL DE AUDIO....28

1. 6 COEFICIENTES DE PREDICCIÓN LINEAL....36

1.7 CUANTIFICACIÓN VECTORIAL....36

1.8 MODELOS OCULTOS DE MARKOV (HMM)....43

1.9 REDES NEURONALES....47

CAPITULO II....49

DISEÑO E IMPLEMENTACIÓN....49

2.1 SOFTWARE....49

2.2 HARDWARE....49

CAPITULO III....84

PRUEBAS EXPERIMENTALES....84

3.1 DESCRIPCIÓN FÍSICA DEL DISPOSITIVO....84

3.2 RESULTADOS OBTENIDOS. 88

3.4 ALCANCES Y LIMITACIONES. 91

CONCLUSIONES Y RECOMENDACIONES 94

BIBLIOGRAFÍA Y ENLACES 98

ANEXOS. 99

INDICE DE FIGURAS

Figura 1. 1. Aparato Fonador 3

Figura 1. 2 Onda de desplazamiento en la glotis 5

Figura 1. 3 Movimiento de las cuerdas vocales durante el ciclo glótico. A. Vista superior. B. Corte coronal. 6

Figura 1. 4 Modelo “source-filter” 7

Figura 1. 5 Sistema de producción de voz 8

Figura 1. 6 Etiología del habla. 12

Figura 1. 7 Micrófono Laringófono Motorola 15

Figura 1. 8 Componentes de la tarjeta DSP TMS320C6416 DSK 16

Figura 1. 9 Diagrama de conexión de la Tarjeta DSP TMS320C6416 DSK 17

Figura 1. 10 Diagrama interno de la tarjeta DSP TMS320C6416 DSK 17

Figura 1. 11 Entorno del Code Composer Estudio 19

Figura 1. 12 Menú Debug del Code Composer Estudio 19

Figura 1. 13 Localización de la librería en la que se encuentra la tarjeta C 6416 DSK 21

Figura 1. 14 Configuración en Simulink del ADC de la tarjeta C 6416 DSK 21

Figura 1. 15 Bloques de extracción y guardar datos de la memoria del DSP 22

Figura 1. 16 Bloques de filtros que la tarjeta puede manejar. 23

Figura 1. 17 Programación del DSK para adquirir datos de audio 23

Figura 1. 18 Generación de Código C 24

Figura 1. 19 Visualización del proceso en el Workspace. 25

Figura 1. 20 Ventana Hamming de 60 muestras 34

Figura 2. 1 Micrófono Laringófono. 49

Figura 2. 2 Cápsula piezoeléctrica correspondiente al laringófono. 50

Figura 2. 3 Adaptación del laringófono como micrófono. 51

Figura 2. 4 TMS320C64X diagrama de bloques del núcleo. 52

Figura 2. 5 Diagrama de Bloques DSP. 53

Figura 2. 6 Ruta de Datos CPU TMS320C64X. 54

Figura 2. 7 Diagrama de bloques para el reconocimiento de voz. 55

Figura 2.8 Modelo de la técnica Homomórfica 60

Figura 2. 9 Modelo Coeficientes Cesptrales 61

Figura 2. 10 Elementos de un VQ. 68

Figura 2. 11 Diagrama de Voronoi 71

Figura 2. 12 Procedimiento de cuantificación para la señal de voz. 72

Figura 2. 13 Representación de un mínimo y un máximo. 73

Figura 2. 14 Representación del vecino más cercano 74

Figura 2. 15 Selección bloque C6416DSK..... 78

Figura 2. 16 Librería C6416DSK Board Support..... 78

Figura 2. 17 Direccionamiento de la Entrada..... 79

Figura 2. 18 Direccionamiento de la Salida..... 79

Figura 2. 19 Bloques de Programación..... 80

Figura 2. 20 Selección de Tiempo de Muestreo..... 80

Figura 2. 21 Selección del tipo de dispositivo. 81

Figura 2. 22 Archivo del sistema de la tarjeta. 81

Figura 2. 23 Paleta de configuración para la compilación..... 82

Figura 2. 24 Compilación del programa. 82

Figura 2. 25 Interacción entre MatLab y Code Composer Studio. 83

Figura 3. 1 Dispositivo de reconocimiento de voz para personas con discapacidad en el habla..... 86

Figura 3. 2 Puerto de alimentación del dispositivo. 87

Figura 3. 3 Puerto de comunicación entre el dispositivo y el computador. ... 87

Figura 3. 4 Entrada de audio Mic In del Dispositivo. 87

Figura 3. 5 Salida del dispositivo de reconocimiento de voz..... 88

Figura 3. 6 Dispositivo conectado a la señora María 88

Figura 3. 7 Dispositivo conectado al señor Juan..... 89

Figura 3. 8 Dispositivo conectado al señor Fernando..... 90

INDICE DE ECUACIONES

Ec 1. 1 Función de Transferencia del filtro digital 32

Ec 1. 2 Filtro pre-énfasis. 32

Ec 1. 3 Ventana Hamming. 33

Ec 1. 4 Cepstrum, coeficientes de predicción lineal. 36

Ec 1. 5 Distancia Euclidiana. 38

Ec 1. 6 Modelos Ocultos de Markov. 43

Ec 1. 7 Redes Neuronales. 48

Ec 2. 1 Energía. 56

Ec 2. 2 Energía promedio. 56

Ec 2. 3 Pre-énfasis. 57

Ec 2. 4 Segmentación. 57

Ec 2. 5 Ventana de Hamming. 58

Ec 2. 6 Modelo Homo mórfica. 60

Ec 2. 7 Modelo Homo mórfico en el dominio algorítmico. 61

Ec 2. 8 Amplitud espectral. 61

Ec 2. 9 Coeficientes cepstrales. 62

Ec 2. 10 coeficientes cepstrales con frecuencia en escala de Mel 62

Ec 2.1 1 Representación Ortogonal Polinomial..... 63

Ec 2.1 2 Medida de la Distancia..... 64

Ec 2.1 3 Regiones de Voronoi. 71

Ec 2.1 4 Calculo de distancia minima. 74

Ec 2.1 5 Calculo de la distancia máxima..... 75

Ec 2.1 6 Moda de datos agrupado. 76

Ec 2.1 7 Calculo de la moda 76

Ec 2.1 8 Clase modal Inferior. 77

Ec 2.1 9 Clase modal Superior. 77

RESUMEN

El presente proyecto está enfocado a la creación de un dispositivo de reconocimiento de voz de las palabras pronunciadas por las personas con discapacidad en el habla que se ha implementado en el DSP TMS320C6416 DSK.

Para ello se ha almacenado una base de datos, de la cual se han obtenido las características representativas de cada palabra a través de los coeficientes Cepstrum.

La adquisición de los datos se realizó a través de un micrófono laringófono, el cual detecta la vibración de las cuerdas vocales, evitando así el ruido exterior. A esta señal se la procesa mediante el uso de Matlab, con filtros especiales, para depurarla y así lograr una mayor confiabilidad de los resultados obtenidos.

Para la identificación de la palabra, se comparó la palabra adquirida con la base de datos almacenada de cada persona, se realizó cuatro procesos para la identificación, los cuales se mencionan en el desarrollo del proyecto: cálculo de la moda, cálculo de mínimos y máximos, cálculo de centroides, medición de la distancia euclidiana.

El procesamiento de la señal se lo realizó en MATLAB, versión 7.4 R2007.

La programación del dispositivo se lo realizó a través de Simulink; a través del toolbox, EMBEDDED TARGET FOR TC 6000 DSP.

CAPITULO I

FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCION

La voz es el sustrato en el que se apoya el método de comunicación habitual del ser humano, con el que se transmite la cultura, con el que se expresan los sentimientos y las emociones. Por su cotidianidad muchas veces pasa desapercibida su extraordinaria importancia, sin embargo por su carácter específico y exclusivamente humano ha sido estudiado desde los inicios de nuestra civilización.[1]

1.2 FUNDAMENTOS DEL HABLA

1.2.1 Descripción del problema médico

Un "trastorno del habla y lenguaje" se entiende como un problema en la comunicación oral. Estos trastornos varían desde simples substituciones de sonido hasta la inhabilidad de comprender o utilizar el lenguaje o el mecanismo motor-oral necesario para el habla y la alimentación. Las alteraciones del habla y la comunicación aparecen con mucha frecuencia en los niños, por lo que es necesario realizar un temprano diagnóstico para que el logopeda pueda trabajar con ellos lo antes posible, evitando así riesgos innecesarios. Causas Algunas causas de los trastornos del habla y lenguaje incluyen la pérdida auditiva, trastornos neurológicos, lesión cerebral, retraso mental, abuso de drogas, impedimentos tales como labio leporino, y

[1] http://www.google.com.ec/url?sa=t&source=web&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fwww.unav.es%2Frevistamedicina%2F50_3%2F1.HISTORIA%2520DE%2520LA%2520VOZ.pdf&ei=3H2vTPWeCcKAlAfT35nlDw&usg=AFQjCNEWQaIuWTnF8Dfsei7a3q9XV6S_IA

abuso o mal uso vocal. Trastorno por déficit de atención. Dificultades en el aprendizaje. Autismo. Esquizofrenia. Parálisis cerebral. Paladar hendido. Lesión en las cuerdas vocales. Síndrome del Maullido del gato (cri-du-chat). Síndrome de Gilles de la Tourette. El desarrollo tardío del habla es uno de los síntomas más comunes en los niños con retraso en el desarrollo y se presenta entre un 5 a 10% de casos. Los niños tienen una probabilidad 3 ó 4 veces mayor que las niñas de presentar trastornos del habla.[2]

1.2.2 Morfología del Aparato Fonador

El proceso básico de producción de la voz es el mismo para hablar y cantar. El cerebro envía señales a través del sistema nervioso central a los músculos de la laringe, cuello y tórax acompañado de un flujo de aire a través del tracto fonatorio obteniendo finalmente la voz.

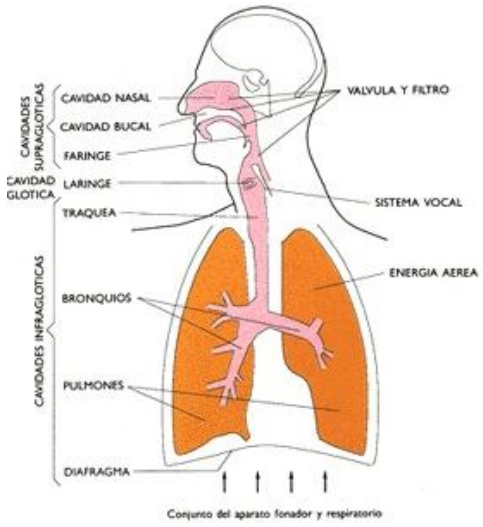

Figura 1. 1. Aparato Fonador

[2] http://www.fundacionbelen.org/base_datos/habla.htm

La voz se define estrictamente como la producción de sonidos por las cuerdas vocales, por un proceso de conversión de energía aerodinámica, la cual es generada en el tórax, el diafragma y la musculatura abdominal, a una energía acústica originada en la glotis. El principio fundamental en la producción de la voz es la vibración de las cuerdas vocales, debido a un acoplamiento y modulación del flujo de aire que pasa a través de ellas generando su movimiento. La eficacia en la transformación de energía está dada por la tensión y la configuración glótica. Hablar se definiría como el resultado del sonido generado en la laringe y modificado por la resonancia de las estructuras supraglóticas. Las teorías que definen el proceso de la voz son: la teoría mioelástica aerodinámica y la teoría cuerpo cubierta. La teoría mioelástica hace alusión al control neuromuscular de la tensión y la elasticidad de las cuerdas vocales. Durante la fonación las cuerdas vocales se encuentran en aducción, tensas contraídas creando una presión subglótica que genera una fuerza ascendente que moviliza las cuerdas vocales en sentido lateral. La teoría aerodinámica se basa en tres principios fundamentales, el primero es el paso de aire de una región de mayor presión a una de menor presión, el segundo hace referencia a la presión de aire la cual disminuye a medida que aumenta la velocidad del aire y finalmente la velocidad del aire aumenta a medida que disminuye el diámetro de la vía por la que fluye. El proceso se inicia cuando las cuerdas vocales se encuentran en aducción y forman un conducto supra y subglótico. El aire generado por los pulmones se acumula en la subglotis llevando a una diferencia de presión con la supraglotis produciendo una fuerza ascendente que abre las cuerdas vocales. El movimiento originado se debe a la propagación del movimiento

vibratorio por el paso del aire a través de la glotis extendiéndose a toda la estructura desde el borde inferior de la glotis al borde superior

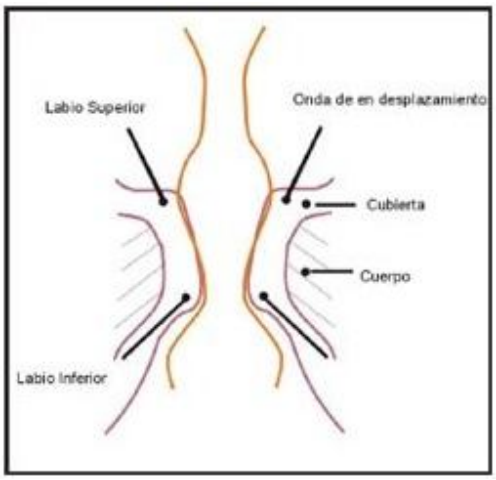

Figura 1. 2 Onda de desplazamiento en la glotis

Posteriormente se combinan varias fuerzas para llevar nuevamente al cierre glótico. La primera se basa en el principio de Bernoulli, donde se genera una presión negativa al paso del aire desplazando medialmente las cuerdas vocales, desde el borde inferior al borde superior de los pliegues. La segunda es una fuerza pasiva de las cuerdas que tiene como función devolver las estructuras a su posición. Finalmente la tercera, el escape de aire glótico que produce disminución de la presión subglótica reposicionando los tejidos, cerrando la glotis y creando nuevamente una presión subglótica alta y el inicio de un nuevo ciclo glótico

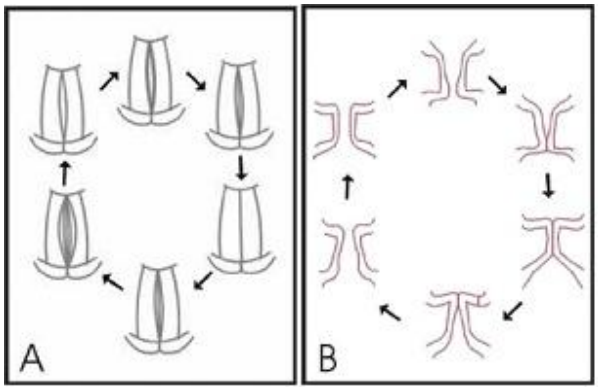

Figura 1. 3 Movimiento de las cuerdas vocales durante el ciclo glótico. A. Vista superior. B. Corte coronal.

La teoría cuerpo cubierta determina las características estructurales de las cuerdas vocales identificando dos capas principales. Una cubierta, formada por epitelio y la capa superficial de la lámina propia, la cual es flexible y elástica. El cuerpo conformado por la capa intermedia y profunda de la lámina propia y el músculo, es rígido y posee propiedades contráctiles. La base fundamental en el funcionamiento de la cuerda vocal en la producción de la voz, se basa en el acoplamiento de la cubierta con el cuerpo en la contracción muscular. La propagación de movimiento vibratorio de apertura y cierre de la mucosa de las cuerdas vocales desde el borde inferior al borde superior, se llama Onda mucosa. El proceso de generación de la voz es complejo y secuencial, requiere un acoplamiento de componentes estructurales, flujo de aire y presión, los cuales crean como producto final la voz, la cual es modificada por la interacción de las características intrínsecas de las cuerdas, la

función pulmonar y las estructuras de resonancia de la vía aérea superior.[3]

1.2.3 Modelo General de producción de la voz.

A continuación se tratara uno de los posibles modelos que se utilizan para caracterizar la producción vocal y las técnicas con las que pueden ser aproximados. Modelo “source-filter” En él se considera a la voz producida por una señal de excitación en forma de impulsos que provienen de la acción de las cuerdas vocales, alternado de forma aleatoria con ruido blanco, que alimenta a un filtro de características variables aunque con una constante de tiempo mucho más lenta.

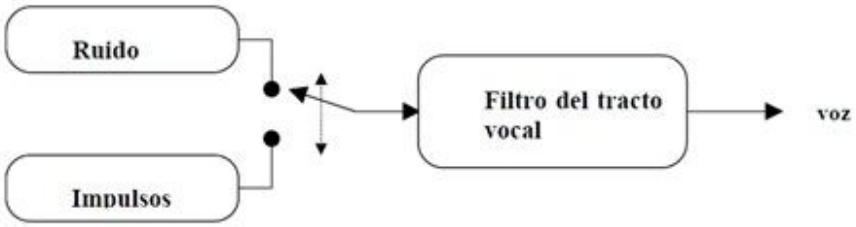

Figura 1. 4 Modelo “source-filter”

De esta manera se puede incorporar al modelo la información de cuerdas vocales más la información del tracto vocal, cuyas piezas móviles están consideradas con las características variables del filtro. La información del tracto vocal está contenida en la envolvente del espectro resultante.[4]

Los órganos que conforman el sistema de producción de voz son:

- Pulmones: Fuente
- Laringe: Contiene las cuerdas vocales
- Cavidad faríngea y cavidad oral, agrupadas en el tracto vocal

[3] http://www.encolombia.com/medicina/otorrino/otorrinosupl31203-contenido.htm
[4] http://www.secyt.frba.utn.edu.ar/gia/IA1_IntroReconocimientoVoz.pdf

- Cavidad nasal (tracto nasal)

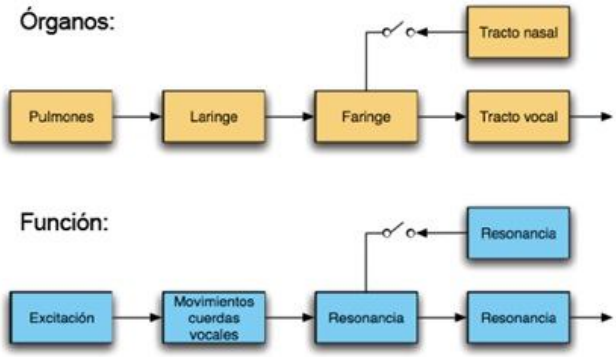

Figura 1. 5 Sistema de producción de voz

1.2.4 Patologías del habla

1.2.4.1 Trastorno del habla

Un "trastorno del habla y lenguaje" se entiende como un problema en la comunicación oral. Estos trastornos varían desde simples substituciones de sonido hasta la inhabilidad de comprender o utilizar el lenguaje o el mecanismo motor-oral necesario para el habla y la alimentación. Las alteraciones del habla y la comunicación aparecen con mucha frecuencia en los niños, por lo que es necesario realizar un temprano diagnóstico para que el logopeda pueda trabajar con ellos lo antes posible, evitando así riesgos innecesarios.

1.2.4.2 Causas

Algunas causas de los trastornos del habla y lenguaje incluyen la pérdida auditiva, trastornos neurológicos, lesión cerebral, retraso mental, abuso de drogas, impedimentos tales como labio leporino, y abuso o mal uso vocal. Trastorno por déficit de atención. Dificultades en el aprendizaje. Autismo. Esquizofrenia.

Parálisis cerebral. Paladar hendido. Lesión en las cuerdas vocales. Síndrome del Maullido del gato (cri-du-chat). Síndrome de Gilles de la Tourette. El desarrollo tardío del habla es uno de los síntomas más comunes en los niños con retraso en el desarrollo y se presenta entre un 5 a 10% de casos. Los niños tienen una probabilidad 3 ó 4 veces mayor que las niñas de presentar trastornos del habla.

Las patologías del habla, serían aquellas que incapacitan a una persona para articular de forma correcta y emitir mensajes hablados. La problemáticas es muy amplia, con lo que los colectivos susceptibles de poder beneficiarse de sistemas de producción de habla artificial son numerosos. En este apartado se introducira una breve clasificación de los distintos trastornos del lenguaje, para dar una idea de la magnitud del problema y de la multitud de factores que intervienen. Dada la diversidad de trastornos que aparecen asociados a distintos orígenes, se ha procedido a clasificarlos de la siguiente manera[5]:

- Trastornos del habla: dislalias; disglosias; disartrias/anartrias; rinolalias; disprosodias.
- Dislalias. Son los trastornos que en mayor número padecen los escolares. Las dislalias parten de una dificultad para alcanzar una correcta articulación. Muy frecuentemente no está asociada a patología alguna por lo que logrando un funcionamiento adecuado en el proceso de aprendizaje y desarrollo del lenguaje queda totalmente recuperada.
- Disglosias. Alteraciones de la articulación por malformación de los órganos del habla. Ejemplos son los niños con fisura

[5] http://www.secyt.frba.utn/INSN0506-TemaProduccionDeHabla-JMG-v46 .pdf

palatina y/o labio leporino. Es imprescindible la intervención clínica para recomponer el órgano dañado antes que una intervención logopedia.

- Disartrias. Alteraciones motrices del habla, es decir, provocadas por lesiones del sistema nervioso que generan trastornos en el movimiento de los músculos que intervienen en la fonoarticulación. Se clasifican de la siguiente manera: fláccida, espástica, atáxica, hipocinética, hipercinética o mixtas. Las hay que son periféricas, paréticas, de tipo cerebeloso o relacionada a alteraciones del tono de los músculos fono articulatorios.
- Anartrias: se trata de un problema de articulación. Su característica principal es la incapacidad total o casi total para producir lenguaje. Es la pérdida del arte de combinar movimientos de los órganos articulatorios.
- Rinolalias: es un trastorno de la voz y del habla relacionado con una alteración en la articulación de algunos fonemas o ausencia total de ellos; se produce una nasalización cuando va unida al timbre nasal de la voz; hipernasalización e hiponasalización cuando hay una mala disposición de las cámaras de resonancia.
- Disprosodias. Es la alteración en el ritmo de la palabra como consecuencia de una disminución o incremento grande en la velocidad del habla, que puede tener un origen neurótico. Esto origina que los fonemas se pronuncien con gran rapidez, con lo que las palabras salen a "tropiezos", con repeticiones, etc. Algunos autores la incluyen dentro de las disfemias.

- Trastornos de la comunicación: disfemia o tartamudez; mutismo; farfulleo; afonías histéricas.
- Tartamudez o disfemia. Es un trastorno relacionado con la fluidez del habla y da lugar a una repetición rápida de sonidos o sílabas, provocando bloqueos al intentar pronunciar una palabra. No está considerada como una enfermedad y en la mayoría de los casos desaparece espontáneamente. Se conocen varios tipos: clónica, tónica o mixta.
- Farfulleo. Se trata de una alteración en la fluidez del lenguaje, en la que el individuo habla a gran velocidad, articulando desordenadamente.
- Mutismo. Relacionado con la negativa por parte de los niños a hablar en determinadas situaciones sociales, incluida la escuela. Estos niños tienen capacidad tanto para hablar como para entender el lenguaje hablado. Suelen utilizar el lenguaje en casa o en ambientes familiares.
- Afonías histéricas. Se relaciona con la pérdida de la voz por un choque afectivo, por el denominado como "miedo escénico", miedo a hablar o a cantar, trauma psíquico por persistencia de alteración vocal, bloqueos por condiciones de stress, o por llamar la atención, sin que exista ninguna alteración anatómica ni funcional en la laringe. Es un trastorno de origen psicológico, se empieza hablando de forma brusca y poco a poco la voz se apaga, hasta terminar susurrando.

1.2.4.3 Definiciones de discapacidades del habla y lenguaje.

La Asociación Americana del Habla, Lenguaje y Audición define un trastorno de la comunicación como: “discapacidad para recibir, transmitir, procesar y comprender, el lenguaje y / o del habla. " La ley IDEA sostiene que los estudiantes con dificultades de comunicación como una “discapacidad del habla y lenguaje” son elegibles para Educación Especial si tienen "un trastorno de la comunicación, como el tartamudeo, la articulación afectada, un trastorno del lenguaje o un deterioro de voz, que afectan negativamente el rendimiento educativo del niño.[6]

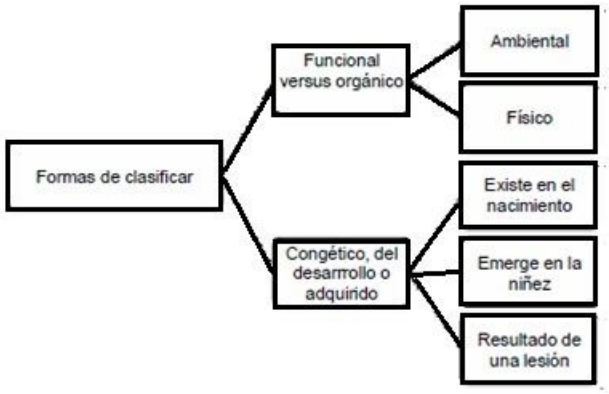

Figura 1. 6 Etiología del habla.

1.2.4.4 Clasificación de las discapacidades del habla y lenguaje

- Trastornos del habla
    - Articulación
    - Fluidez
    - De voz

[6] http://mayramartinezplana.net/files/11086-10660/Habla_y_lenguaje.pdf

- Trastorno de procesamiento central auditivo (CAPD –siglas en ingles)
    - Dificultad en procesar sonidos
- Trastornos del lenguaje
    - Trastorno fonológico
    - Apraxia del lenguaje
    - Trastorno morfológico
    - Trastorno semántico
    - Déficits sintácticos
    - Dificultades con la pragmática

1.3 FUNDAMENTOS DE LA ADQUISICIÓN DE DATOS.

1.3.1. ¿Qué es el sonido y cómo se captura en un computador digital?

El sonido es una percepción humana que permite obtener gran cantidad de información de nuestro entorno. El fenómeno físico que lo produce es el movimiento del aire, o mejor dicho, el movimiento de una onda de presión (onda acústica).El órgano del oído es un sistema muy sofisticado en el que se capta la onda acústica (oído externo y medio), se descompone frecuencialmente y se convierte en estímulos eléctricos (cóclea), se transmite al cerebro (nervio auditivo), y se procesa (cerebro) para construir la percepción subjetiva que se llama sonido. La capacidad del cerebro para procesar sonidos es increíble y se está lejos todavía de ser capaces de imitarlo. Para capturar el sonido se utilizan micrófonos, que convierten la onda acústica (movimiento) en una señal eléctrica, y para generar sonido se utilizan altavoces, que realizan la operación contraria, convirtiendo la señal eléctrica en una onda acústica. Para

poder ser usada en un computador digital la señal eléctrica procedente del micrófono debe ser digitalizada. Para ello, primero se muestrea y luego las muestras se cuantifican y codifican.

Los parámetros fundamentales de la digitalización son: la frecuencia de muestreo, en muestras por segundo (hercios), y el número de bits empleado para codificar cada muestra. El resultado es una secuencia de códigos binarios manejable en un computador digital. Cuando el sonido es estéreo, hay dos señales (una por cada canal, izquierdo y derecho) que se digitalizan por separado.

1.3.2. Micrófono Laringófono

El laringófono es un micrófono especialmente sensible a las vibraciones de la laringe, y por consiguiente, de gran utilidad en todos aquellos casos en que se quiera comunicar en ambientes muy ruidosos, ya que sólo capta las vibraciones de la laringe y no los ruidos externos.

Además de su uso para personas con problemas de voz, los laringófonos han tenido aplicación en todos esos ambientes ruidosos, donde era necesario comunicarse y que los destinatarios de sus mensajes los recibieran de forma comprensible, a pesar de ser transmitidos desde entornos muy ruidosos: dentro de la cabina de un avión antiguo, en la cabina de un camión, para motoristas, etc., donde el ruido del motor es bastante fuerte. También pueden usarse en ambientes más silenciosos pero donde no se puede tener un micrófono a mano: submarinismo, parapente, etc.

Estos micrófonos especiales que, aplicados al cuello, captan la voz producida en las cuerdas vocales, casi no se fabrican hoy en día por

falta de demanda, y puede ser bastante difícil dar con alguna cápsula microfónica para laringófonos.[7]

Figura 1. 7 Micrófono Laringófono Motorola

Sistemas de Funcionamiento del Laringófono

- El sistema VOX le permite tener las manos libres, se activa por las cuerdas vocales. Muy aconsejado en situaciones de mucho ruido, pues no capta el ruido de ambiente, solo las vibraciones de las cuerdas vocales.
- El sistema PPT. Este sistema es el más conocido y más utilizado, para hablar hay que pulsar el PTT.

1.3.3. DSP TMS320C6416 DSK De Texas Instruments

El TMS320C6416 es un DSP (Procesador de Señales Digitales) diseñado y fabricado por la compañía Texas Instrument, surgiendo con algunas ventajas sobre el resto de los procesadores. La principal diferencia entre los DSP´s y los modernos procesadores, es que estos se diseñan para ser escalables; es decir para que puedan operar en paralelo con otros dispositivos similares. Muchos de los

[7] http://www.gcnlevante.com/radiocom/uso_libre/motorola/accesorio_motorola__laringofono_x20m.htm#

procesadores se engloban dentro de la filosofía CISC, Aunque se pueden encontrar en el mercado algunos que operen bajo la filosofía RISC; estos últimos dedicados para aplicaciones concretas como la telefonía móvil. Un DSP es increíblemente más rápido, procesa datos en tiempo real y es ideal para aplicaciones que no toleran retraso. Un buen ejemplo de estas aplicaciones se puede encontrar en el diseño de los últimos aparatos que realizan electrocardiogramas, dado que se requieren tomar determinaciones (por el médico) antes de que cambie la señal a la entrada (corazón del paciente). Las aplicaciones más comunes y económicas se encuentran en la telefonía celular.[8]

1.3.3.1 Componentes que conforman la Tarjeta

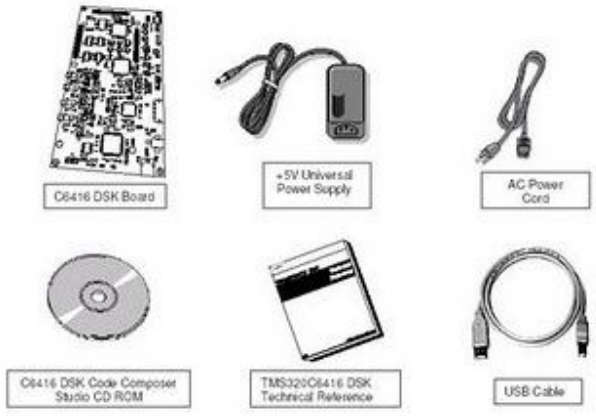

Figura 1. 8 Componentes de la tarjeta DSP TMS320C6416 DSK

- Tarjeta C6416 dsk.
- Fuente de alimentación 5v.
- Cable de AC.
- Code Composer Studio.
- Cable USB.

[8] http://edwsoft-dspti.blogspot.com/2009/05/dsp-tms320c6416-dsk-de-texas.html

1.3.3.2 Diagrama de conexión de la Tarjeta

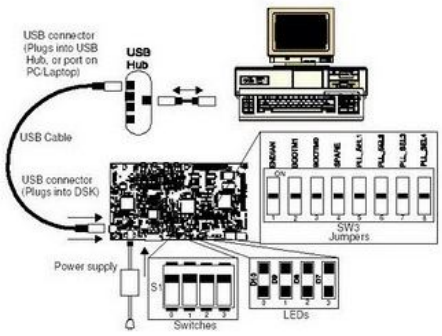

Figura 1. 9 Diagrama de conexión de la Tarjeta DSP TMS320C6416 DSK

En la Figura 1.9 se puede observar el diagrama de conexión de la tarjeta con la computadora para poderla programar. Como se puede observar se conecta el cable de AC con la Fuente de 5 V y luego esta con el conector ubicado en la tarjeta, de esta forma la tarjeta ya queda energizada, luego se conecta el cable USB con la tarjeta y luego con el puerto del PC. De esta forma ya queda lista la tarjeta para ser simulada y programada desde el Code Composer o desde el Matlab.

1.3.3.3 Diagrama de la Tarjeta

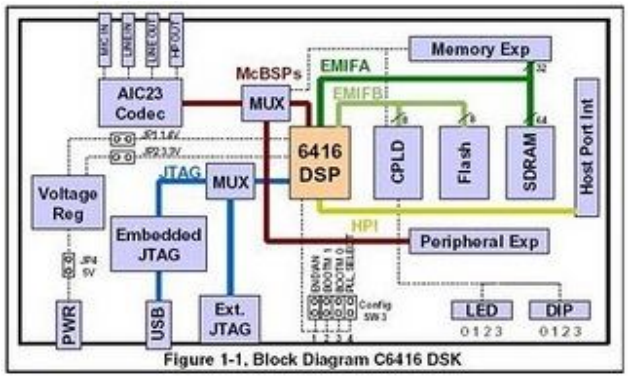

Figura 1. 10 Diagrama interno de la tarjeta DSP TMS320C6416 DSK

En la Figura 1.10 se puede ver el diagrama de bloques de la tarjeta en él se puede observar, todos sus componentes principales. El AIC23 Codec es la parte que permite enlazar la tarjeta a aplicaciones de audio para procesar digitalmente este tipo de señales estos plug son stereo de 35 mm, posee dos líneas de entrada la Mic In y Line In al igual que dos de salida Line Out y Hp Out que son los conversores DAC y ADC de la DSP. El bloque JTAG es el que hace la interfaz USB de la tarjeta y a través de este también me permite emular los sucesos programados desde la PC. También se ve en el diagrama la parte principal de la tarjeta DSP 6416 que es el componente principal que controla todos los demás bloques, junto a este se encuentran las expansiones; la PCI que se puede utilizar para diversas aplicaciones y enlazar otro tipo de hardware con la tarjeta tales como memorias, tarjetas de adquisición de dato, etc. La expansión para memorias externas, la expansión de periféricos donde de allí se puede obtener salidas y entradas digitales. La tarjeta también posee interfaz HPI, leds indicadores, interruptores programables. Estos son en sí lo bloques más generales que posee la tarjeta en hardware.

1.3.4 Entorno de Desarrollo CODE COMPOSER STUDIO

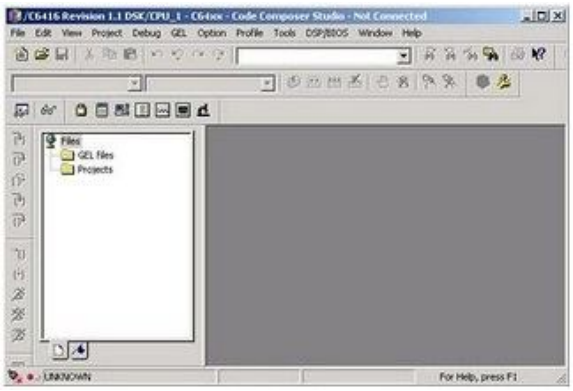

Figura 1. 11 Entorno del Code Composer Estudio

Este es el entorno de desarrollo donde se programa la tarjeta, el lenguaje de programación es C, un lenguaje universal de programación, también fusionado en ocasiones en ensamblador. Desde el software se digita el código y a su vez se programa la tarjeta. Para que el software detecte la tarjeta, primero se hace la conexión USB, a continuación, desde el menú Debug se hace click en Connect (ver Figura.1.12.)

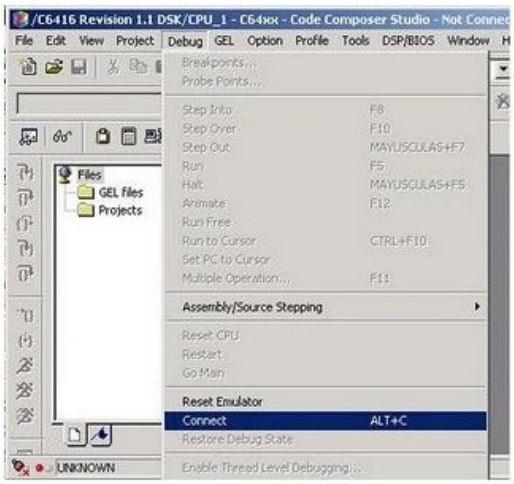

Figura 1. 12 Menú Debug del Code Composer Estudio

Una vez que el software detecta la tarjeta y está apta para ser programada; se crea un nuevo proyecto y un archivo en *.C, desde allí se incluye las librerías que se van a utilizar; esto lo hace el programa por defecto. Después se compila el proyecto, y se programa la tarjeta con el archivo de extensión .out, el cual se busca en el menú File , Load Program; desde allí se abre la carpeta Debug del proyecto y se elige el archivo del proyecto, en ese instante es cuando se pasa el programa a la tarjeta, luego se corre(Run) el programa y la tarjeta empieza a funcionar según lo programado, esto se conoce cuando el led Busy de la tarjeta se encuentra encendido; la ejecución del programa se suspende desde el Code Composer.

1.3.5 Programación de la Tarjeta usando Matlab

Una de las grandes ventajas de usar esta tarjeta es la compatibilidad que tiene con el simulink de Matlab, ya que dispone de los Toolbox idóneos, que están diseñados específicamente para este tipo de tarjetas, y hace que la programación sea más sencilla, y que se puedan crear lazos de control y diagramas en bloque requeridos, que luego el compilador que usa matlab transforma todo este tipo de diagrama en un lenguaje en C para que el compilador del Code Composer lo ajuste y lo programe en la tarjeta. En la figura 1.13 se observa la toolbox específica para esta tarjeta, para localizarla se busca en el menú Embedded Target for TC 6000 DSP y luego se selecciona la opción C6416 DSK que es la tarjeta que se utiliza en este proyecto. Al abrir esta librería se observan varios bloques de la tarjeta como se indica en la Figura 1.13.

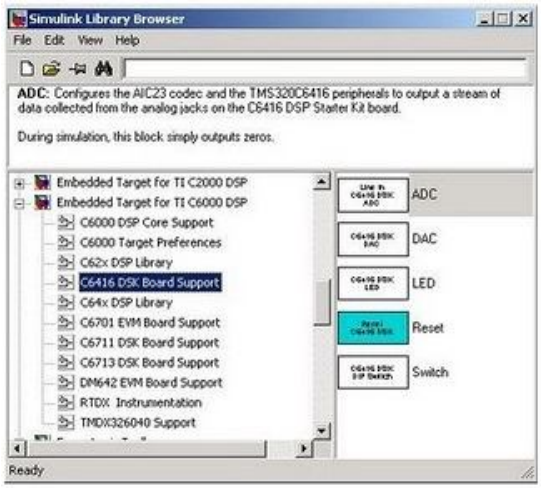

Figura 1. 13 Localización de la librería en la que se encuentra la tarjeta C 6416 DSK

A continuación se presenta el ADC, es el conversor análogo digital, que en hardware corresponde a las dos entradas análogas estéreo abriendo este bloque y configurando sus propiedades, se puede seleccionar el canal a utilizar, sea desde Line In o Mic in; la frecuencia de muestreo; el tamaño en bits de la conversión; el tipo de dato a retornar; la cantidad de tramas por muestreo.

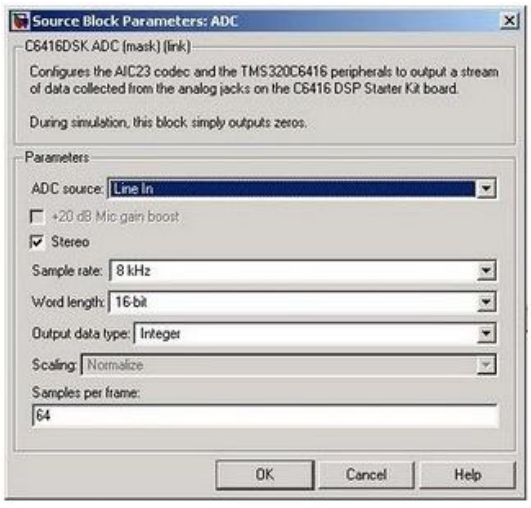

Figura 1. 14 Configuración en Simulink del ADC de la tarjeta C 6416 DSK

También está el DAC que es el conversor Digital Análogo, el cual en Hardware hace referencia a las dos salidas digitales, que al igual que el ADC se puede configurar sus parámetros, como la entrada a utilizar, el tamaño en bits de los datos digitales a convertir el tipo de datos a recibir, y la frecuencia de muestreo. El siguiente bloque comprende el manejo de LEDS que la tarjeta hace referencia a cuatro leds indicadores, en estos también se configura el tipo de dato a recibir, El bloque DIP SWITCH corresponde a los cuatro switch que contiene la tarjeta estos actúan como cuatro entradas digitales, allí también se configura el tipo de dato a retornar (ver Figura.1.14). Otro bloque es *reset*, que permite el reinicio de la tarjeta, dando click sobre este cuando el programa en la DSP se está ejecutando. Los anteriores bloques hacen referencia a las partes físicas del DSP. Aparte de estas librerías también hay otras las cuales son de funciones ya internas de la DSP como la de guardar en memoria datos al igual que extraer de ella como se observa en la Figura 1.15.

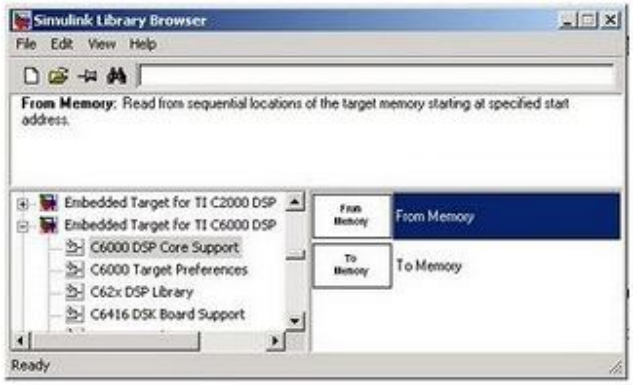

Figura 1. 15 Bloques de extracción y guardar datos de la memoria del DSP

Hay otras librerías o bloques compatibles con varias tarjetas de la serie C6000 al cual pertenece la DSK6416, allí se puede encontrar filtros ya listos para su uso tales como, FIR, FFT (Transformada

Rápida de Fourier) como se ve en la figura 1.16 estos bloques se obtienen desde el menú C64X DSP Support

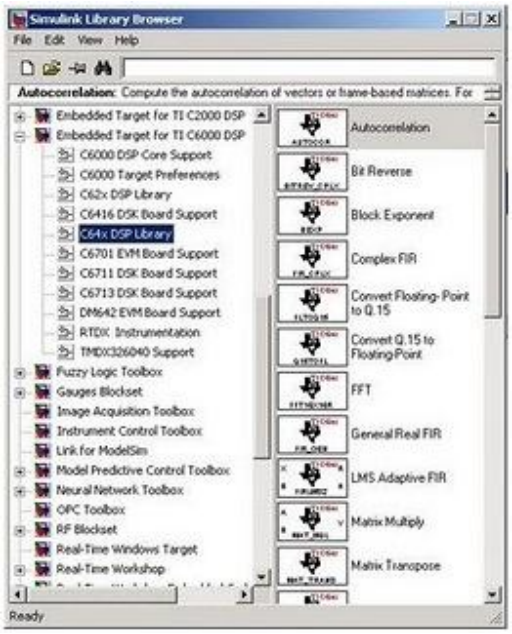

Figura 1. 16 Bloques de filtros que la tarjeta puede manejar.

Todos los bloques anteriormente mencionados, pueden interactuar con otros toolbox del Simulink, haciendo de esta una herramienta aun más poderosa y eficiente.

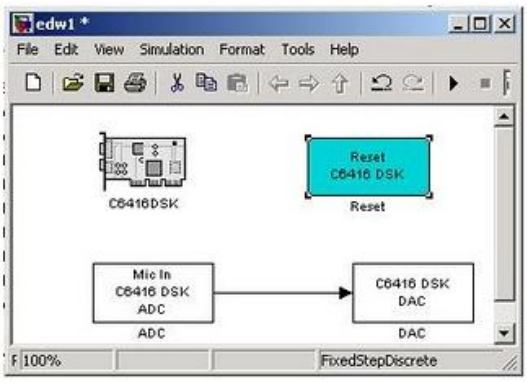

Figura 1. 17 Programación del DSK para adquirir datos de audio

En la figura 1.17 se observa un ejemplo sencillo de una aplicación para la DSP, después de tener el modelo ya listo, se configura en matlab para habilitar el driver que reconozca la tarjeta y pueda hacer el enlace con el Code Composer, esto se hace en el menú Simulation, Configuration Parameters

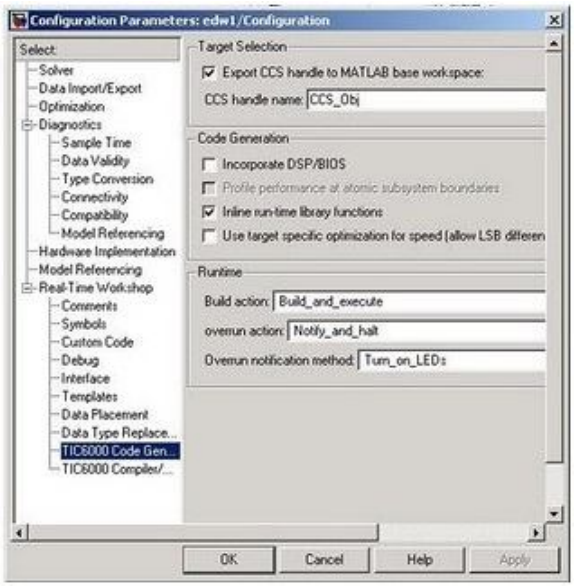

Figura 1. 18 Generación de Código C

En la Figura 1.18 se observa la ventana que permite la configuración, donde se tiene que seleccionar el modo discreto, también se busca el driver para controlar la tarjeta que se está trabajando, se debe seleccionar el código al que debe transformar el modelo en este caso en código C. En la casilla Build Action se selecciona lo que el Matlab hace a la hora de compilar el modelo , aquí se encuentran varias opciones, solo crear proyecto en Code Composer, a crearlo y compilarlo hasta hacer todo el proceso completo y llegar a cargar el programa a la tarjeta sin necesidad de hacerlo desde el Code Composer.

Después de compilar en el Workspace de Matlab se ve todo el proceso completo ver figura 1.19

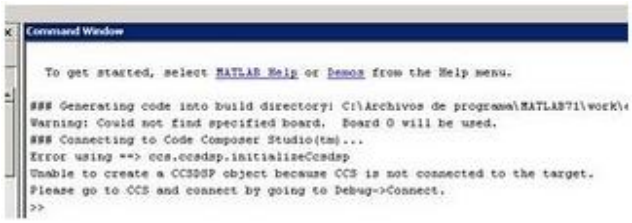

Figura 1. 19 Visualización del proceso en el Workspace.

1.4 SOFTWARE MATLAB

MATLAB (abreviatura de MATrix LABoratory) es una potente herramienta para el tratamiento matemático de datos en el computador. Las funciones de MATLAB pueden ejecutarse interactivamente mediante comandos, o utilizarse para escribir programas. En este proyecto se utilizó el toolbox y una GUIDE.

MatLab es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros de Control en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa. MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y

visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

MATLAB está siendo utilizado como herramienta de investigación para la resolución de complejos problemas planteados en la realización y aplicación de modelos matemáticos en ingeniería. Los usos más característicos de la herramienta se encuentran en áreas de computación y cálculo numérico tradicional, prototipaje algorítmico, teoría de control automático, estadística, análisis de series temporales para el proceso digital de señal.

MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyo especializados, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc.

Además también se dispone del programa Simulink que es un entorno gráfico interactivo con el que se puede analizar, modelizar y simular la dinámica de sistemas no lineales.[9]

[9] http://www.usc.es/gir/docencia_files/tdd/tutorial_matlab.pdf

1.4.1. Signal Processing Toolbox Matlab

Tiene una gran colección de funciones para el procesamiento de señal en el Signal Processing Toolbox. Este incluye funciones para:

- Análisis de filtros digitales incluyendo respuesta en frecuencia, retardo de grupo, retardo de fase.
- Implementación de filtros, tanto directo como usando técnicas en el dominio de la frecuencia basadas en la FFT.
- Diseño de filtros IIR, incluyendo Butterworth, Chebyschev tipo I, Chebyshebv tipo II y elíptico.
- Diseño de filtros FIR mediante el algorítmo óptimo de Parks-McClellan.
- Procesamiento de la transformada rápida de Fourier FFT, incluyendo la transformación para potencias de dos y su inversa, y transformada para no potencias de dos.

1.4.2. Comandos De Matlab Para Procesamiento De Señales

- Time response:
    - dimpulse - Discrete unit sample response.
    - dinitial - Discrete initial condition response.
    - dlsim - Discrete simulation to arbitrary inputs.
    - dstep - Discrete step response. filter - SISO z - transform simulation.
    - impulse - Impulse response.
    - initial - Continuous initial condition response.
    - lsim - Continuous simulation to arbitrary inputs.
    - ltitr - Low level time response function.
    - step - Step response.
    - stepfun - Step function.

- Frequency response:
    - bode - Bode plot (frequency response).
    - dbode - Discrete Bode plot (frequency response).
    - dnichols - Discrete Nichols plo t.
    - dnyquist - Discrete Nyquist plot.
    - dsigma - Discrete singular value frequency plot.
    - fbode - Fast Bode plot for continuous systems.
    - freqs - Laplace -transform frequency response.
    - freqz - Z-transform frequency response.
    - ltifr - Low level frequency response function. margin - Gain and phase margins.
    - nichols - Nichols plot.
    - ngrid - Draw grid lines for Nichols plot.
    - nyquist - Nyquist plot.
    - sigma - Singular value frequency plot.

1. 5 PROCESAMIENTO DE LA SEÑAL DE AUDIO[10]

1.5.1. Muestreo y Cuantificación

1.5.1.1. Muestreo

La etapa de muestreo consiste en convertir la señal analógica, continua en el tiempo, en una señal discreta en el tiempo. El principio fundamental del muestreo es el denominado teorema de Nyquist. El cual enuncia que si la frecuencia de muestreo es mayor o igual al doble del ancho

[10] http://www.google.com/url?sa=t&source=web&cd=2&ved=0CBgQFjAB&url=http%3A%2F%2Fwww.gnewbook.org%2Faction%2Ffile%2Fdownload%3Ffile_guid%3D70523&rct=j&q=DISE%C3%91O%20E%20IMPLEMENTACI%C3%93N%20DE%20UN%20PROTOTIPO%20DE%20RECONOCIMIENTO%20DE%20VOZ%20BASADO%20EN%20MODELOS%20OCULTOS%20DE%20MARKOV%20PARA%20COMANDAR%20una%20silla%20de%20ruedas%2Bpdf&ei=7QiITfnfKMux0QHvhsHfDQ&usg=AFQjCNEkXdh0tg5rrB2jhYHt0QBVnZ3aZw

de banda de la señal a muestrear, se podrá recuperar la señal en su totalidad mediante una interpolación basada en funciones seno. Este hecho es importante, ya que nos indica que si el muestreo se realiza de forma correcta, no se pierde información.

Los estudios sobre las características de las señales de voz han demostrado que la mayor parte de la información necesaria para la inteligibilidad del habla se encuentra por debajo de los 4 KHz. De hecho el ancho de banda disponible tradicionalmente en las líneas telefónicas es algo menor de 4 KHz. Aunque hay que destacar que algunos sonidos emitidos por el aparato fonador poseen frecuencias mucho más elevadas, por ejemplo los sonidos fricativos (sonidos que se producen cuando un articulador se acerca a una zona de articulación de modo que el paso del aire se obstruye parcialmente, produciendo una fricción) los cuales pueden alcanzar los 10 KHz., pero la pérdida de esta información no supone un déficit sustancial en la información del habla.

Según Nyquist se sabe que es necesaria una frecuencia de muestreo de por lo menos el doble del ancho de banda de la señal a caracterizar, sobre esta base y para un análisis mínimo (en lo que respecta a frecuencia) de la señal de voz se utiliza una frecuencia de muestreo fs de 8khz, aunque se suele usar 16khz si se desea obtener mayor detalle en frecuencia lo que mejora la resolución para tratamiento de la señal. La cuantificación más comúnmente usada, es de 8 bits, mínimo requerido para una calidad baja, puede mejorarse su S/R con una técnica no lineal de cuantificación,

se obtienen excelentes resultados aumentando la cuantificación a 16 bits.

La siguiente etapa será aquella que se encargue de amplificar las señales a niveles que sean manejables. A partir de la señal analógica obtenida se hace necesario convertir la señal a formato digital para poder procesarla en la computadora lo que se realiza mediante dos procesos: muestreo y cuantificación. La señal vocal tiene componentes de frecuencia que pueden llegar a los 10 kHz., sin embargo la mayor parte de los sonidos vocales tiene energía espectral significativa hasta los 5 kHz solamente los sonidos fricativos poseen componentes que pueden llegar a los 10 kHz.

1.5.1.2. Cuantificación

En la cuantificación el valor de cada muestra de la señal se representa como un valor elegido de entre un conjunto finito de posibles valores. Se conoce como error de cuantificación (o ruido), a la diferencia entre la señal de entrada (sin cuantificar) y la señal de salida (ya cuantificada), interesa que el ruido sea lo más bajo posible. Para conseguir esto y según sea la aplicación a desarrollar, se pueden usar distintas técnicas de cuantificación:

- Cuantificación uniforme
- Cuantificación logarítmica
- Cuantificación no uniforme
- Cuantificación vectorial

Cuantificación uniforme: En los cuantificadores uniformes o lineales la distancia entre los niveles de reconstrucción es siempre la misma, la mayoría usan un número de niveles

que es una potencia de 2. No hacen ninguna suposición acerca de la señal a cuantificar, de allí que no proporcionen los mejores resultados. Pero son los más fáciles y menos costosos a implementar.

Cuantificación logarítmica: Para evitar desperdicio de niveles de reconstrucción y de ancho de banda se utiliza un método sencillo para mejorar el incremento de la distancia entre los niveles de reconstrucción conforme aumenta la amplitud de la señal. Para conseguir esto se hace pasar la señal por un compresor logarítmico antes de la cuantificación. Esta señal comprimida puede ser cuantificada uniformemente. A la salida del sistema la señal pasa por un expansor. A esta técnica se le llama compresión.

Cuantificación no uniforme: Este cuantificador utiliza la función de la distribución de probabilidad, conociendo esto se puede ajustar los niveles de reconstrucción a la distribución de forma que se minimice el error cuadrático medio. Cuantificación vectorial: Este método cuantifica los datos en bloques de N muestras. En este tipo de cuantificación, el bloque de N muestras se trata como un vector N-dimensional.

1.5.2. Pre-Procesamiento

La etapa de pre-énfasis se realiza con el propósito de suavizar el espectro y reducir las inestabilidades de cálculo asociadas con las operaciones aritméticas de precisión finita. Además se usa para compensar la caída de -6 dB que experimenta la señal al pasar a través del tracto vocal. Se usa un filtro digital de primer orden cuya función de transferencia es:

$$H(z) = 1 - az$$

Ec 1. 1 Función de Transferencia del filtro digital

El filtro de pre-énfasis tiene como objetivo el hecho de que la señal de voz tiene un contenido más elevado en bajas que en altas frecuencias. La siguiente expresión hace que el espectro presente un aspecto más plano:

$$yn = sn - axsn - 1$$

Ec 1. 2 Filtro pre-énfasis.

1.5.3. Ventaneo

La señal de voz es un proceso aleatorio y no estacionario. Esto supone un inconveniente a la hora de analizar la señal, no obstante es posible salvar este problema si se tiene en cuenta que a corto plazo de tiempo (del orden de ms) la señal es casi-estacionaria esto da lugar a un tipo de análisis donde se obtienen segmentos o tramas de señal de pocos ms denominado análisis localizado. A este proceso donde se obtienen tramas o segmentos consecutivos de señal se le denomina enventanado.

El enventanado requiere que cada una de las tramas sea multiplicada por una función limitada en el tiempo de tal manera que su valor fuera de su intervalo sea nulo.

El entramado de la señal se puede considerar como la multiplicación de esta por una señal rectangular, lo que en el espacio frecuencial se traduce en convolucionar el espectro de la señal de audio con una sinc. Para evitar en lo posible la aparición de componentes en alta frecuencia, debidas a las

discontinuidades de la señal rectangular, se aplica una ventana de Hamming en el proceso de enventanado de la señal.
De los diferentes tipos de ventana se puede destacar dos: Ventana rectangular y Ventana Hamming.

Ventana Hamming

La utilización de una ventana rectangular tiene el inconveniente que en los extremos de dicha ventana la función decae rápidamente y esto da lugar a que se produzca el fenómeno de Gibbs. Para evitar este fenómeno es necesario una ventana cuya función tenga una caída suave en sus extremos una posible solución es utilizar una ventana Hamming.

En la etapa siguiente, la señal pre-acentuada se toma cada 10 o 20ms. por espacio de 20 o 40 ms. y se la somete a una ventana de Hamming con el objeto de suavizar la señal en los bordes de dicha ventana. Esta es la ventana que generalmente se usa para el análisis de señales de voz, y se define como:

$$wnT = 0.54 - 0.46cos2nN \qquad 0 \leq n \leq N$$

Ec 1. 3 Ventana Hamming.

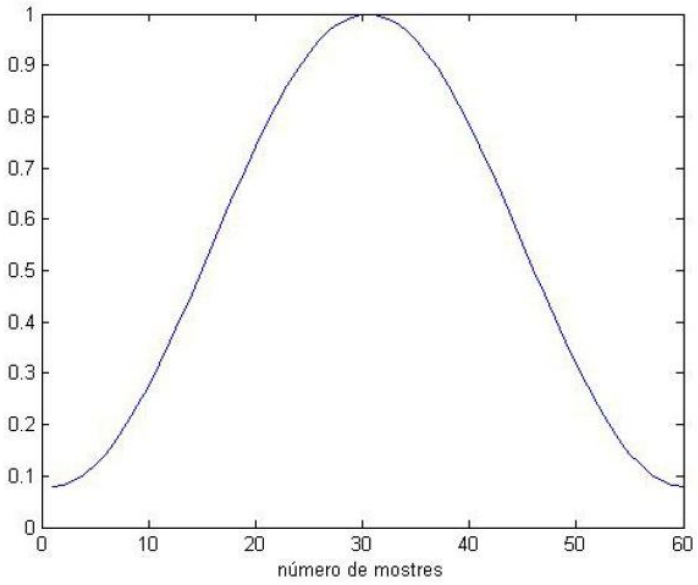

Figura 1. 20 Ventana Hamming de 60 muestras

Al utilizar una ventana Hamming es preciso tener en cuenta que las muestras en los extremos de la ventana sufrirán una ponderación a diferencia de las muestras de la zona central cuyo valor no experimentara ningún cambio, para compensar este efecto de ponderación se hace necesario un solapamiento de las ventanas de tal manera que el desplazamiento de la ventana sea inferior a la longitud de esta.

1.5.4. Segmentación

En el reconocimiento de señales de voz, se hace necesario determinar con adecuada precisión los puntos de inicio y final de cada palabra, es decir, se debe diferenciar las partes de señal que llevan información de voz de aquellas que no llevan voz. Este procedimiento evita gastar memoria y tiempo de cálculo en las tramas que no contienen información evitando así obtener resultados erróneos en el análisis de las señales de voz. Se han

planteado diferentes maneras de clasificar eventos en una señal de voz. Una opción simple y más empleada, está relacionada con la generación de la voz e incluye tres estados: silencio cuando no hay voz; sonoro cuando se presenta vibración de las cuerdas vocales; y sordo cuando las cuerdas vocales no vibran. En general, un sistema de clasificación de voz puede tener inconvenientes en distinguir un fonema sordo de baja amplitud y corta duración, del silencio; o un fonema sonoro de baja amplitud de un fonema sordo o incluso del silencio. El error a su vez aumenta a medida que la relación señal a ruido disminuye.

El problema de encontrar los puntos de inicio y fin de palabra es fundamental en procesamiento de voz. Por ejemplo, en reconocimiento automático de palabras aisladas, es necesario encontrar las regiones de la señal que corresponden a cada palabra a ser analizada. De la correcta segmentación de la señal depende en gran medida la exactitud del proceso de reconocimiento. De hecho, las fallas en la segmentación de la señal constituyen una de las principales fuentes de error en los sistemas de reconocimiento de voz, ya que algunos sonidos que pueden captarse, correspondientes a ruido de fondo, podrían eventualmente confundirse con voz; por ejemplo, el espectro de la respiración tiene semejanzas con el de un fonema fricativo. La detección de los límites de palabra también se realiza con el fin de evitar cálculos innecesarios, al procesar únicamente las partes de la señal que corresponden a voz

1. 6 COEFICIENTES DE PREDICCIÓN LINEAL

Para el reconocimiento de señales de voz la información relevante es la relativa al tracto vocal, ya que es la que define el tipo de sonido que se ha emitido. Por el contrario, la información relativa a la excitación no es útil, ya depende de factores altamente variables como la entonación, sexo del locutor, estado emocional del locutor, etc.

Por ello, una buena manera de representar la información relativa exclusivamente al tracto vocal es mediante un vector de parámetros que contenga los primeros L coeficientes cepstrales $(c(1); c(2); \dots\dots\dots; c(L))$, siendo L un número pequeño (Típicamente entre 8 y 20). El primer coeficiente cepstral c (0) tampoco se suele incluir en el vector, ya que está relacionado con la energía de la señal, que es también un parámetro sometido a una alta variabilidad. El cepstrum LPC proporciona mejores resultados que el cepstrum FFT, por lo que es preferible su uso. Este cepstrum puede obtenerse mediante la siguiente recursión:

$$c(n) = \begin{cases} 0 & n \leq 0 \\ -a_1 & n = 1 \\ -a_n - \sum_{k=1}^{n-1} \frac{k}{n} c(k) a_{n-k} & n > 1 \end{cases}$$

Ec 1. 4 Cepstrum, coeficientes de predicción lineal.

1.7 CUANTIFICACIÓN VECTORIAL

Una parte importante en cualquier tipo de procesamiento de voz es la optimización de los algoritmos en cuanto a velocidad y almacenamiento, entonces, la cuantificación de vectores trae consigo la idea de clasificar un conjunto de vectores, luego de lo cual se

buscarán los mejores representantes para reducir el tamaño de la información a manejar. La forma de medir la fidelidad de un cuantificador es determinar el error que éste produce al reemplazar los datos de entrada que recibe por los vectores representantes o codewords, dicho parámetro es llamado error por distorsión. La finalidad de un cuantificador es obtener un conjunto de vectores representativos llamado codebook, que presente el menor error por distorsión, por ejemplo para cuantificar los vectores de observación.

Ventajas

Reduce el almacenamiento de la información de análisis. Se reduce el cálculo para determinar distancias entre vectores espectrales. La representación del VQ se limita a una tabla que contiene las distancias entre pares de vectores del codebook.

Desventajas

Distorsión en la representación del vector. Hay un número finito de vectores en el codebook, el proceso de “elección" del mejor representante es equivalente a cuantificar el vector y conduce a un cierto nivel de error de cuantificación. De cualquier modo con cualquier codebook finito siempre habrá un nivel de ruido o error. El almacenamiento requerido para los vectores del codebook no es pequeño. Cuanto más grande sea el codebook menor es el error. Para un codebook de 1000 o más entradas, el almacenamiento no es irrelevante. Hay que realizar un balance entre error de cuantificación, procesamiento y almacenamiento del codebook. Componentes de un cuantificador vectorial Para construir un cuantificador vectorial se necesita:

a) Un gran número de vectores de observación, V1, V2,....., Vn, que conforman el grupo de entrenamiento. El grupo de entrenamiento se usa para crear el grupo de vectores del

codebook "optimo" que representa la variabilidad espectral observada en el grupo de entrenamiento.

b) Una medición de distancia entre cada par de vectores espectrales de observación para agrupar el conjunto de vectores de entrenamiento como así también para asociar o clasificar vectores arbitrarios a cada entrada del codebook.
c) Un procedimiento de clasificación para ubicar y calcular los centroides. Sobre la base del particionamiento que clasifica el grupo de n vectores en M clústeres o sectores primero se elige el número M, codewords del codebook, para luego proceder a la clasificación.

1.7.1. Distancia Euclidiana

Mide la línea recta que une dos puntos en un espacio Euclidiano. Si se toma como ejemplo un espacio unidimensional la distancia será la resta de ambas coordenadas:

$$D(x_1, x_2) = x_2 - x_1$$

Ec 1. 5 Distancia Euclidiana.

En un plano será la hipotenusa del triángulo rectángulo formado por los puntos (Pitágoras) en tres dimensiones y por extensión en un espacio multidimensional de orden n, se calcula una "hipotenusa" n dimensional, lo que generaliza el cálculo de la distancia mínima.

1.7.2. Vectores de Observación

Al final de los distintos pasos para el tratamiento de la señal de voz, se obtiene un vector que contiene la información vocal que representa a la ventana temporal correspondiente, de alguna manera una colección de características que describen de la

mejor manera posible la voz humana. Estos vectores son conocidos en la literatura del reconocimiento de voz como vectores de observación. Cabe aclarar que existen varias formas de representación de estas características como LPC (Linear Prediction Code) o Auditory System, pero la que en la actualidad da los mejores resultados es el análisis Cepstral, en particular los coeficientes MFCC (Mel Frecuency Cepstral Coeficients). También suele incorporarse al vector de Observación la información de la primera y segunda derivadas del Cepstrum con respecto al tiempo para agregar información de las características dinámicas del sistema y el logaritmo de la energía total de la ventana.

1.7.3. Clasificación de Vectores

El objetivo de un módulo clasificador es agrupar una cantidad de vectores característicos, N, en una cantidad M ($M < N$), discreta, de sectores o celdas de clasificación logrando que las características en cada sector sean similares. Existen muchos criterios para lograr dicho objetivo y a continuación se verá algunos de los más comunes. Imagine que la media multidimensional de un determinado sector i, es μi($con\, 1 < i < M$), y a continuación ingresa al clasificador un vector de observación o, se puede clasificar dicho vector calculando la "distancia" a la que se halla de cada una de las M medias y asignándolo al sector más "cercano". Este método de clasificación se denomina k-Means debido a que se agrupan los vectores en torno a k valores medios, quedando formados k sectores (en nuestro caso $k = M$). Existe el problema de

inicialización de los valores de μi, y su reestimación a medida que progresa el algoritmo.

Algoritmos de Clasificación

Se puede decir, en general, que los N vectores originales de tamaño D quedarán representados por M vectores, cada uno de los cuales es llamado "palabra de código" o codeword (Cw), el grupo entero de dichos vectores, forma un "libro de códigos" o codebook, quedan entonces delimitadas M regiones o sectores, llamados regiones de Voronoi.

Los principales algoritmos de clasificación de vectores son descritos a continuación:

Algoritmo K-Means

a) Inicialización: Arbitrariamente se elige M vectores o palabras de código, codewords, como el grupo inicial del codebook.
b) Búsqueda del más cercano: Por cada vector de observación, se busca el codeword en el codebook que es el más cercano (en términos de distancia), y asigna a ese vector a la celda correspondiente.
c) Actualización del centroide: actualiza el codeword en cada celda o sector usando el centroide de los vectores de entrenamiento asignados a un sector.
d) Iteración: Repite los pasos a y c hasta que la distancia media caiga debajo de un umbral prefijado.

La forma de cada sector o celda o partición es muy dependiente de la medida de distorsión espectral y las estadísticas de los vectores en el grupo de entrenamiento. Este método es el más

simple y por tanto existen numerosas modificaciones y mejoras, algunos de sus puntos débiles son:

a) Los resultados dependen en forma muy acentuada de los valores iníciales elegidos como palabras de código.
b) También hay gran dependencia del número de sectores M así como de la implementación de la "distancia" usada.
c) Puede suceder que algunos de los sectores resulten vacíos.

Algoritmo LBG

Se analizará con algún detalle debido a su buen desempeño, para eso se comenzara por el algoritmo fundamental LBG. El algoritmo LBG, lleva su nombre debido a sus autores Y. Linde, A. Buzo y R. M. Gray, en él se elige un codeword inicial de entre los vectores de datos a clasificar, luego se utiliza el algoritmo de división binaria para duplicar el número de codewords, los vectores de observación se agrupan en torno a los codewords que les presentan menor distancia, se recalculan los codewords como la media multidimensional de cada sector y se agrupan nuevamente los datos, el proceso se detiene cuando el codebook no presenta variación significativa y al llegar al número de codewords deseados. Este algoritmo de gran popularidad (que utiliza el algoritmo k-Means) produce codebooks que logran un mínimo local en la función de error por distorsión. Para generar un codebook de M sectores o palabras de código: En primer lugar designando un codeword inicial para luego utilizando una técnica de división llegar a obtener un codebook inicial, luego iterando la misma técnica de división en los codewords hasta que se llega a obtener el número de codewords igual a M que va a ser el tamaño del codebook deseado.

El procesamiento se denomina división binaria:

a) Designar 1 vector del codebook o codeword inicial, éste resulta ser el centroide del grupo de los vectores de entrenamiento.
b) Calcular la media del grupo de entrenamiento:
    1. Calcular el error o distancia media entre el codeword inicial y los vectores de entrenamiento:
c) Duplicar el tamaño del codebook mediante la división de cada codeword.
d) Usar el algoritmo K-Means para tomar el mejor grupo de centroides para la separación del codebook.
e) Iterar pasos c y d hasta llegar a un codebook de tamaño M. Una de las causas que motivo el uso de un VQ fue la suposición que, en el límite, el codebook debería idealmente tener 36 vectores, uno por cada fonema, suposición que es incorrecta.

1.7.4. Utilización del cuantificador y del codebook

Una vez construido el codebook, el procedimiento para cuantificar vectores es básicamente realizar una búsqueda completa a través del codebook para encontrar el mejor representante. Si se anota los vectores del codebook, de tamaño M, como $Cw, 1 \leq w \leq M$, y se toma al vector de observación a ser cuantificado como V, luego el vector representante o codeword, Vm*,.

Un procedimiento de cuantificación para señal de voz elige el vector más cercano del codebook al vector de observación y utiliza ese vector denominado codeword, como la representación resultante para etapas posteriores. Se refiere como al vector "vecino" más cercano, toma como entrada, vectores de señal de

voz y da como respuesta, a su salida, el vector que mejor representa esa entrada.

1.8 MODELOS OCULTOS DE MARKOV (HMM)

1.8.1. Elementos de un HMM

Dado un modelo HMM pueden definirse sobre el mismo los siguientes elementos, que caracterizan completamente al HMM:

- N: cantidad de estados del modelo. Si bien no son observables, para algunas aplicaciones suelen tener algún significado físico asociado.
- M: el número de símbolos de observación distintos por estado. Corresponde a la salida física del sistema modelado.

$$\lambda = A = a_{ij} \quad 1 \leq i,j \leq Na_{ij} = Pqt + 1 = jqt = i \quad con\, j$$
$$= 1Na_{ij} = 1 \,\forall\, iB = bjk \quad 1 \leq i \leq N,$$
$$1 \leq k \leq Mbjk = POt = kqt = j\ con\, k = 1Mbjk = 1\ \forall\, j\pi$$
$$= \pi i \quad 1 \leq i \leq N\pi i = Pq1$$
$$= i\ \ prob.del\ estado\ inicial, con\ i = N\pi i = 1$$

Ec 1. 6 Modelos Ocultos de Markov.

Siendo los elementos aij comúnmente expresados por medio de la matriz de transiciones A, matriz de probabilidad de transición entre estados y los elementos bjk comúnmente expresados por medio de la matriz B, matriz de probabilidad de emisión de símbolos en un estado cualquiera. A partir de una secuencia de observaciones se puede inferir el modelo dinámico más probable λ, resultando un modelo para el proceso deseado. Por lo tanto, un HMM puede describirse como un modelo generativo que modela

un proceso (en nuestro caso la voz) como una sucesión de estados que se conectan por transiciones.

Cada estado tiene asociada una salida de una observación, con su correspondiente distribución de probabilidades. Cada transición está asociada a una probabilidad que la caracteriza. El modelo lleva el nombre de Markov debido a su restricción de que la probabilidad de un estado en el tiempo actual, sólo depende del estado previo, y oculto ya que los estados no son observables en forma directa, sino a través de sus observaciones que son los vectores correspondientes, característicos de la señal. La probabilidad de dicha salida modela la imposición acústica y la probabilidad de transición entre estados modela su duración.

1.8.2. HMM y problemas asociados

Desde el punto de vista del modelado de segmentos o símbolos de voz, los HMM son muy versátiles, pudiéndose realizar el modelo de fonemas, palabras, y hasta frases enteras. Provisto por el cuantificador de vectores el HMM tendrá una cantidad discreta de observaciones M y también una cantidad N de estados. Los problemas que se deben enfrentar son:

- En primer lugar es necesario obtener el modelo $\lambda(A, B, \pi)$pi) para un determinado grupo de observaciones que haga óptimo el calculo de $P(O \setminus \lambda)$, es decir, que haga máxima la probabilidad de haber generado dichas observaciones, proceso conocido como entrenamiento del HMM.

- En segundo término calcular las probabilidades $P(O\backslash\lambda)$, es decir que la secuencia de observaciones O, haya sido generada por un dado modelo λ.
- Por último si se tiene un grupo de observaciones O, y el modelo λ, calcular la secuencia de estados óptima, la que hace máxima $P(secuencia\backslash \mathrm{O}, \lambda)$.

Si se quiere encarar el problema de reconocimiento de palabras aisladas y su número no hace que computacionalmente sea irrealizable, puede calcularse el modelo (λ) de cada una de ellas, con entrenamiento adecuado, y luego al llegar al sistema un conjunto de observaciones desconocido, calcular las probabilidades $P(O\backslash\lambda i)$y decidir por el modelo de mayor probabilidad. Si el sistema es de reconocimiento continuo o el número de palabras es grande, lo anterior es imposible, ya sea el hecho de tener modelos de todas las palabras posibles como el de calcular tal número de probabilidades. Una alternativa a esto es tratar de obtener la secuencia de estados óptima, hecho que permite el ahorro de tiempo de cálculo.

1.8.3. Entrenamiento de HMM

El proceso de entrenamiento de un modelo HMM, utiliza los algoritmos necesarios para realizar el cálculo o estimación de los parámetros que definen al modelo. Se trata básicamente de un proceso iterativo que maximiza en forma local la probabilidad de que una secuencia de observación haya sido generada por un modelo particular $[P(O\backslash\lambda)]$ y que garantiza en cierta forma la convergencia del proceso a partir de un modelo inicial aleatorio.

Uno de los métodos más conocidos para realizar esta tarea es la técnica de maximización de la Estimación, y como una especialización de la misma, el algoritmo de Baum – Welch. Este último define los parámetros de un HMM como se muestra a continuación:

- Probabilidades de transición de estados (Matriz A)
- Probabilidades de emisión (Matriz B)
- Vector de probabilidad inicial

Entonces, dada una secuencia muestral, se busca, mediante el método de maximización de la estimación, obtener el modelo HMM que tenga más probabilidad de generar la secuencia indicada utilizando las fórmulas expuestas y el algoritmo general que se indica a continuación:

- **Obtención del modelo inicial:** se obtiene en forma totalmente aleatoria, sujeto como es de suponer a las restricciones de probabilidades comunes. Existe la posibilidad de implementar mejoras que ayuden a obtener parámetros iníciales más exactos. Cabe aclarar que cuanto más exacto o cercano al máximo global se encuentre el modelo inicial, más exacto será el modelo final obtenido.
- **Cálculo de P:** la probabilidad de que la observación haya sido generada por el modelo obtenido es calculada con la ayuda de algoritmos intermedios auxiliares que permiten reducir la complejidad computacional del proceso (algoritmos forward y backward). Esta probabilidad es calculada por cada uno de los modelos obtenidos hasta verificar que la misma es máxima.

- **Reestimación del modelo:** se trata de recalcular los parámetros del modelo utilizando las fórmulas anteriores, basándose para ello en el modelo obtenido en la iteración anterior.
- **Modelo óptimo:** una vez alcanzada la máxima probabilidad, se está en presencia del modelo óptimo el cual debe ser guardado para su utilización posterior en lo que se da a llamar el repositorio de modelos.

En resumen, el algoritmo de maximización de la estimación propone obtener en forma iterativa modelos λpara una secuencia de observaciones muestra, e ir comparando las probabilidades de generación de los mismos (las cuales, como es de suponer, son crecientes hasta alcanzar el máximo) hasta detectar que se llegó a un máximo local para dicha probabilidad. Alcanzado este máximo (paso garantizado por la convergencia del método) se toma al modelo final como el que más probablemente pueda generar la secuencia de observaciones caracterizada por la muestra.

Habiendo generado un modelo óptimo para una secuencia de observaciones dada, solo resta determinar cómo interactuar para obtener un reconocedor de palabras aisladas a partir de lo expresado.

1.9 REDES NEURONALES

Las RNA son sistemas de procesamiento de información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. En todo modelo de RNA se tienen cuatro elementos básicos.

- Un conjunto de conexiones, pesos ó sinapsis que determinan el comportamiento de la neurona, las cuales pueden ser excitadoras, presentan un signo positivo (conexiones positivas) y las inhibidoras presentan un signo negativo (conexiones negativas).
- Una función que se encarga de sumar todas las entradas multiplicadas por sus pesos correspondientes.
- Una función de activación que puede ser lineal ó no lineal empleada para limitar la amplitud de la salida de la neurona.
- Una ganancia exterior que determina el umbral de activación de la neurona.

Desde que el psicólogo Frank Rosenblatt en 1957 introdujo el modelo del perceptrón de una sola capa, las RNA se convirtieron en una herramienta poderosa para solucionar diversos tipos de problemas relacionados con la clasificación, estimación funcional y optimización del reconocimiento de patrones. El modelo propuesto se observa en Ec. 1,9, donde $X_{p1}, \ldots \ldots \ldots X_{pi}$ son las unidades de entrada, $W_{j1} \ldots \ldots \ldots W_{ji}$ son los pesos de la RNA, b_i es la ganancia ó umbral de activación, N_{pj} es el producto de los pesos con respecto a la entrada, f es la función de activación de la RNA y finalmente y_{pj} es la salida de la RNA, estas variables se relacionan en la siguiente expresión:

$$y_{pj} = f\left(N_{pj} = \sum_{i=1}^{m} X_{pi} W_{ji} + b_i\right), \qquad para\ m \in IR, m < \infty$$

Ec 1. 7 Redes Neuronales.

CAPITULO II

DISEÑO E IMPLEMENTACIÓN

2.1 HARDWARE

2.1.1. MICRÓFONO LARINGÓFONO.

El sistema Laringófono / auricular con PTT permite hablar aprovechando la técnica de “vibración vocal”, el micrófono se coloca en la parte lateral del cuello y aprovechando las vibraciones de las cuerdas vocales se pueden conseguir comunicación clara y limpia.

Durante el uso del dispositivo, este inhibe los ruidos externos, como la música, o la voz de otras personas que hablan cerca del micrófono, permitiendo el paso de únicamente la señal de voz del discapacitado o el usuario de prueba, sin interferencias significativas.

Figura 2. 1 Micrófono Laringófono.

Función PTT:

Esta opción le permite activar de manera manual la función VOX. Dispone de un botón auxiliar PTT con velcro, para fijarlo cómodamente en la mano o en el dedo.

Además de su uso para personas con problemas de voz, los laringófonos tienen aplicación en todos los ambientes ruidosos, donde es necesario comunicarse y que los destinatarios de sus mensajes los reciban de forma comprensible, a pesar de ser transmitidos desde entornos ensordecedores, como por ejemplo: dentro de la cabina de un avión antiguo, en la cabina de un camión, para motoristas, etc, donde el ruido del motor es bastante fuerte.

El micrófono laringófono detecta la vibración de las cuerdas vocales a través de una cápsula piezoeléctrica.

Figura 2. 2 Cápsula piezoeléctrica correspondiente al laringófono..

El funcionamiento de un laringófono es similar a un teléfono manos libres. Para lo cual se realizó una adaptación por hardware, y poder usarlo como micrófono únicamente. Para realizar esta adecuación, se usó un Jack de auricular

perteneciente a un teléfono celular, del cual se localiza el pin de tierra y el pin de micrófono. (Ver Figura.2.3)

Figura 2. 3 Adaptación del laringófono como micrófono.

El nivel de salida de audio, es estándar, lo cual permite hacer la conexión directa a la tarjeta de reconocimiento TMS320C6416; la adecuación permite disponer de un solo canal monofónico.

2.1.2. KIT DE ENTRENAMIENTO DEL DSP TMS320C6416.

2.1.2.1. Procesador Digital De Señal TMS320C6416.

Es el procesador C64x tiene una velocidad de reloj que puede variar desde 600 a 1100 MHz, la cual se puede incrementar hasta en un 83%, incrementando la velocidad de procesamiento de señales.

El C64x se muestra en la figura 2.4. El núcleo del C6416 consiste de 8 unidades funcionales, 2 archivos de registros, y dos rutas de datos. El software usa datos de 12-16 bits, para diseño de equipos de tercera generación.

La extensión de 16-bits en la unidad funcional de multiplicación está también presente en las otras seis unidades funcionales. Esta incluye operaciones dobles de 16-bits de adición/sustracción, comparador, desplazamiento, max. /min y de valor absoluto. Tipos de dato en paquetes de

8-bits y 16-bits son usados por las herramientas de generación de código para tomar ventaja de esta extensión.

Al duplicar los registros en el archivo de registros y la duplicación del ancho de la ruta del dato, como lo utilizado en la instrucción de empaquetamiento, el compilador del C6000 es mucho más eficiente, por tener menos restricciones.

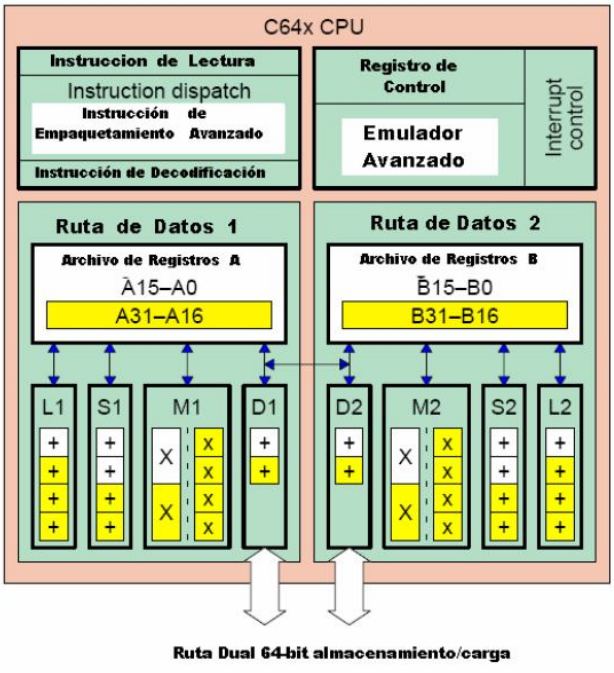

Figura 2. 4 TMS320C64X diagrama de bloques del núcleo.

2.1.2.2. Arquitectura TMS320C6X.

El TMS320C64X tiene una Arquitectura VLIW (256 bits de ancho), para alimentar hasta ocho instrucciones de 32 bits para las ocho unidades funcionales durante cada ciclo de reloj. Incluye dos niveles de memoria interna L1 y L2, el nivel

1 es un L1P Cache de 16 KB y L1D Cache de 16 KB, cada cache nivel 1 (L1P y L1D) está conectado a L2 Cache de 1 MB, unificado para programa y datos. La figura 2.10 muestra el diagrama de bloques del DSP.

Posee dos rutas de datos punto fijo, la ruta A y la ruta B. Cada ruta de datos contiene cuatro unidades de ejecución: ALU, un registro de desplazamiento, un multiplicador, un sumador/restador usado para generar direcciones. Cada ruta de datos también contiene un archivo de registros con 32 registros de propósito general de 32 bit, el doble de lo que tiene TMS320C62X.

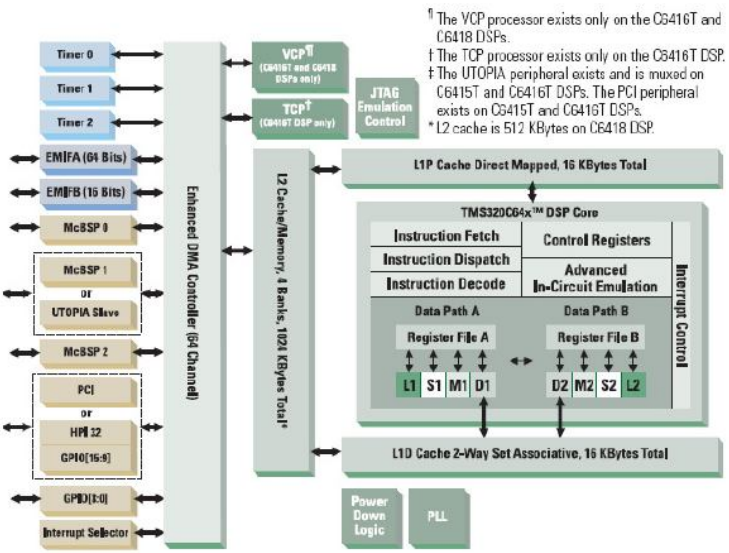

Figura 2. 5 Diagrama de Bloques DSP.

Las ocho unidades de ejecución son capaces de ejecutar más de 8 instrucciones de 32 bits en paralelo utilizando los dos archivos de registro. El TMS320C64X puede operar con datos de 8, 16, 32

y 40 bits de largo y también puede operar con 64 bits (doble palabra), cuando se cargan o almacenan datos desde o hacia la memoria.

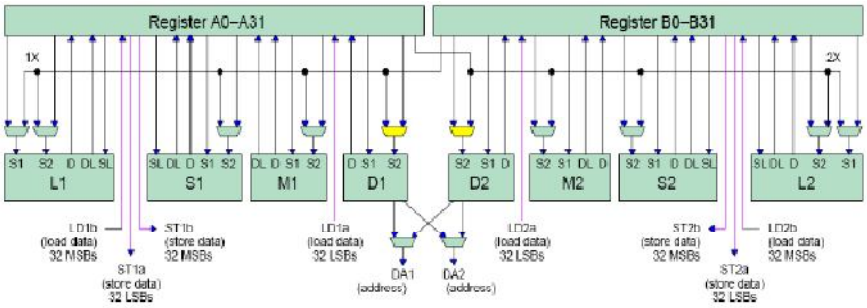

Figura 2. 6 Ruta de Datos CPU TMS320C64X.

El TMS320C64X maneja datos de 64 bits usando un par de registros. Típicamente las unidades ALUs, registros de desplazamiento y la generación de direcciones opera con operando de 32 bits, pero el ALU y los registros de desplazamiento pueden operar con de 40 bits. Los multiplicadores realizan multiplicaciones de 16x16 bit, 8x8 bits. Comparando al TMS320C62X, el TMS320C64X agrega 4 de 8-bit y 2 de 16 bits instrucciones aritméticas SIMD y lógica para algunas de las unidades de ejecución, e introduce nueva instrucción producto punto, con vectores, 4 de 8 bits y 2 de 16 bits, para las dos unidades de multiplicación. Con este desarrollo el TMS320C64X puede realizar multiplicaciones paralelas, cuatro de 16 bits u ocho de 8 bits.

2.2 SOFTWARE.

4.1.1. PROCESAMIENTO DE LA SEÑAL DE VOZ

Para el reconocimiento de voz se ha implementado un algoritmo basado en el siguiente diagrama de bloques:

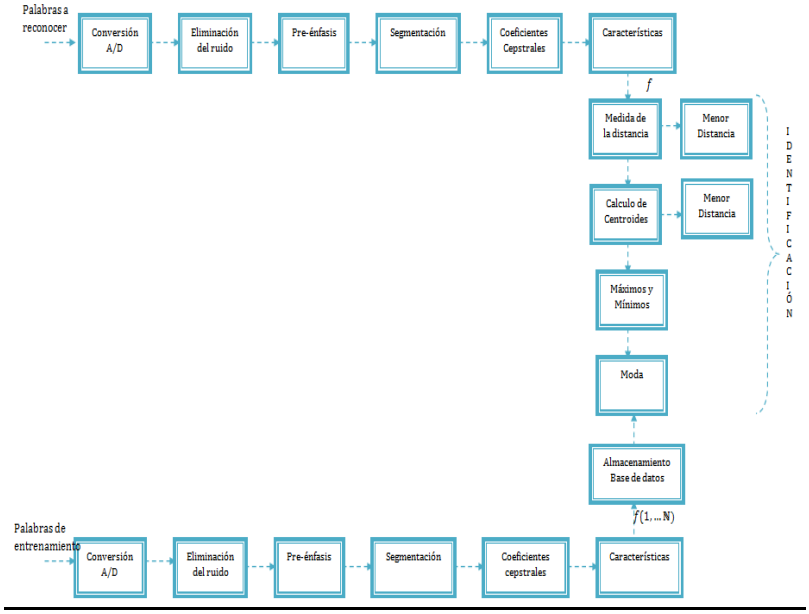

Figura 2. 7 Diagrama de bloques para el reconocimiento de voz.

4.1.1.1. Conversión Analógica Digital.

Un conversor (o convertidor) analógico-digital (CAD), (o también ADC del inglés "Analog-to-Digital Converter") es un dispositivo electrónico capaz de convertir una entrada analógica de voltaje en un valor binario, Se utiliza en equipos electrónicos como ordenadores, grabadores de sonido y de vídeo, y equipos de telecomunicaciones. La señal analógica, que varía de forma continua en el tiempo, se conecta a la entrada del dispositivo y se

somete a un muestreo a una velocidad fija, obteniéndose así una señal digital a la salida del mismo.

Para nuestro caso se ha escogido un micrófono laringófono para la adquisición de la señal de audio.

La conversión análoga digital se ha realizado a través de la tarjeta de procesamiento de señales TMD320C6416.

4.1.1.2. Eliminación del ruido

La señal digitalizada es escaneada y las zonas de silencio son removidas por medio del cálculo de energía en un corto tiempo. Segmentos de 10ms se escogieron para este propósito. En un segmento la energía promedio es menor que un valor de umbral, proporcional a la energía promedio total de la señal. Las siguientes fórmulas se utilizaron:

$$En = \sum_{k=1}^{Wn} |x[k]|^2 w[n-k]$$

Ec 2. 1 Energía.

$$E_{avg} = \frac{1}{N} \sum_{k=1}^{N} |x[k]|^2$$

Ec 2. 2 Energía promedio.

Donde En es la energía promedio de cada segmento y Eavg es la energía promedio de la señal entera. El valor umbral escogido THRES=0.2.

4.1.1.3. Pre-énfasis

Se aplica un filtro digital pasa altas de primer orden a la señal, para enfatizar las frecuencias altas de los formantes por dos razones, primero para que no se pierda información durante la segmentación, porque la mayoría de la información está contenida en las frecuencias bajas, en segundo remueve la componente DC de la señal, aplanando espectralmente la señal. Uno de los filtros de pre-énfasis más utilizados tiene la ecuación:

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

Ec 2. 3 Pre-énfasis.

a = 0.95 en nuestro caso, definido experimentalmente.

4.1.1.4. Segmentación.

La segmentación consiste en cortar la señal en segmentos de análisis. La señal de voz es asumida como estacionaria en estos segmentos.

Durante la segmentación los segmentos son guardados cada uno como la columna de una matriz, para el posterior procesamiento de la señal de voz. Se empleó una ventana de Hamming de 30ms, a la señal de voz, enfatizada previamente con el filtro de pre-énfasis, con un desplazamiento típico 10ms entre cada ventaneo.

Se realiza el algoritmo en base a las siguientes fórmulas:

$$Q_n = \sum_{k=-\infty}^{\infty} x[k]w[n-k]$$

Ec 2. 4 Segmentación.

Qn es cada n^th cuadro de segmentación

$$w[n] = 0.54 - 0.46cos\left\{\frac{2\pi(n)}{N}\right\}$$

Ec 2. 5 Ventana de Hamming.

En la ecuación de la ventana de Hamming, N es el largo de cada cuadro o segmento de análisis.

Para los valores de los parámetros que hacen falta para la implementación del algoritmo, se utiliza la siguiente tabla:

Tabla 2.1. Valores de Parámetros para el reconocimiento de voz.

Parámetro	**Valor**
N número de muestras en el segmento de análisis.	240(30 ms)
M número de muestras entre cada segmento.	80(10 ms)
P LPC orden de análisis.	10
Q dimensión del vector cepstral derivado del LPC	15

4.1.1.5. Características

En el reconocimiento del habla, la señal de voz pre-procesada, ingresa a un nuevo procesamiento para producir una representación de la voz en forma de secuencia de vectores o agrupaciones de valores que se denominan parámetros, que deben representar la información contenida en la envolvente del espectro.

Hay que tener en cuenta que el número de parámetros debe ser reducido, para no saturar la base de datos, porque mientras más parámetros tenga la representación menos fiables son los resultados y más costosa la implementación.

Existen distintos métodos de análisis para la extracción de características, y se concentran en diferentes aspectos representativos. En este caso se analizara los dos de mayor importancia para el análisis de la voz:

- Análisis de predicción lineal (LPC)

- Análisis cepstral

En nuestro caso se utilizar el Análisis Cepstral para la extracción de las características.

Cepstrum

Los sonidos de la voz se pueden representar mediante un espectrograma, que indica las componentes frecuenciales de la señal de voz.

Es así entonces como el espectro de la señal, proporciona información acerca de los parámetros del modelo de producción de voz, tanto de la excitación como del filtro que representa el tracto vocal.

Desde el principio de la década de los 70 los sistemas homo mórficos han tenido una gran importancia en los sistemas de reconocimiento de voz. Estos sistemas homomórficos son una clase de sistemas no lineales que obedecen a un principio de superposición. De estos los sistemas lineales son un caso especial.

La razón para realizar un procesado homomórfico del habla se resume en la Figura 2.2

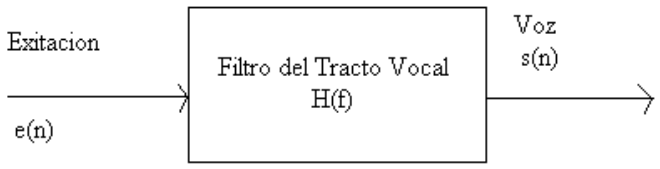

Figura 2.8 Modelo de la técnica Homomórfica

La señal de voz s(n) se descompone en una parte de excitación e(n) y en un filtro lineal $H(e^{j\theta})$como se mencionó anteriormente. Así, en el dominio de la frecuencia se tiene:

$$(|S(e^{j\theta})|) = H(e^{j\theta})E(e^{j\theta})$$

Ec 2. 6 Modelo Homomórfica.

En el dominio logarítmico, por su parte, las dos componentes anteriores pueden separarse empleando técnicas convencionales del procesamiento de señal.

Eso se logra del siguiente modo:

$$log(|S(e^{j\theta})|) = log(|H(e^{j\theta})|) + log(|E(e^{j\theta})|)$$

Ec 2. 7 Modelo Homomórfico en el dominio algorítmico.

Para la mayoría de aplicaciones de voz solamente se necesita la amplitud espectral.

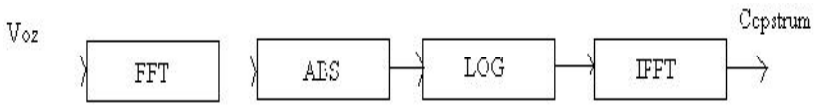

Figura 2. 9 Modelo Coeficientes Cesptrales

En la salida de este sistema se tiene entonces:

$$c(n) = \frac{1}{N_s}\sum_{k=0}^{N_s-1} log|S_{med}(k)|e^{j\frac{2\pi}{N_s}kn} \qquad para\ 0 \le n \le N_s - 1$$

Ec 2. 8 Amplitud espectral.

En cual caso, el valor c(n) se conoce como coeficientes cepstrales derivados de la transformada de Fourier. Ns es el número de puntos con que se calcula la transformada. Esta ecuación puede ser convenientemente simplificada teniendo en cuenta que el espectro logarítmico es una función real simétrica.

$$c(n) = \frac{2}{N_s}\sum_{k=1}^{N_s} S_{med}\big(I(k)\big)cos\left(\frac{2\pi}{N_s}kn\right)$$

Ec 2. 9 Coeficientes cepstrales.

En los cálculos lo habitual es usar solamente los primeros términos ($n \leq 20$), en este proyecto se emplearon los primeros quince, por los resultados obtenidos.

Por otro lado, *I(k)* representa una función que traduce la posición de un valor en frecuencia al intervalo donde este contenido.

Es posible, a la hora de calcular un coeficiente cepstral, transformar el espectro utilizando bandas definidas según escalas de Mel; en cuyo caso a este tipo de parámetro se conoce como coeficientes cepstrales con frecuencia en escala de Mel (*MFCC*).

Partiendo del análisis de predicción lineal también es posible obtener la expresión de los coeficientes cepstrales asociados:

$$c(0) = \log(1) = 0$$

$$c(i) = -\alpha(i) - \sum_{j=1}^{i-1}\left(1 - \frac{j}{i}\right)\alpha(j)c(i-j), \;\; 1 \leq i \leq N_c$$

Ec 2. 10 coeficientes cepstrales con frecuencia en escala de Mel

En el sistema de reconocimiento de voz en MATLAB existe una función para obtener los coeficientes cepstrales utilizando *la FFT.* La función utilizada es la *rceps*, que nos proporciona el cepstrum real de la función ingresada, por medio del algoritmo mostrado en la figura 10; es decir, que es la implementación del algoritmo mostrado anteriormente. La razón principal para utilizar los

coeficientes cepstrales se debe a la ventaja adicional que puede derivarse de ellos, una serie de parámetros que son invariantes sin importar las distorsiones que puedan ser introducidas por el micrófono o por cualquier sistema de transmisión.

Características del Cepstrum:

Los coeficientes son normalizados para reducir variabilidades espectrales durante largos periodos de tiempo. Los coeficientes son expandidos por medio de una representación polinomial ortogonal durante intervalos de 90ms cada 10ms. Este intervalo es adecuado para preservar información de transición entre fonemas. Solamente los dos primeros coeficientes ortogonales polinomiales son utilizados. Las siguientes ecuaciones se utilizan en el algoritmo:

$$P_{0j} = 1$$

$$P_{1j} = j - 5$$

Los primeros dos coeficientes de la representación ortogonal polinomial son:

$$a = \frac{\left(\sum_j^9 xj\right)}{9} \qquad b = \frac{\left(\sum_j^9 x_j P_{1j}\right)}{\left(\sum_j^9 P_{1j}^2\right)}$$

Ec 2.1 1 Representación Ortogonal Polinomial.

Los coeficientes a y b representan el promedio, de la función de tiempo de cada coeficiente cepstral en cada segmento respectivamente. Dicha representación es una función del tiempo de los coeficientes cepstrales $x_t(i)$ y los coeficientes polinomiales

de primer orden que están representados por $b_t(i)$ donde t es el número de segmento e *i* es el índice de los coeficientes cepstrales. Como el valor de p es escogido como 10, la representación resultante es una función del tiempo de 20 elementos de características.

4.1.1.6. Identificación

4.1.1.6.1. Medida de la distancia

Una característica fundamental de los sistemas de reconocimiento es la forma en que los vectores característicos son combinados y comparados con los patrones de referencia.

Para poder realizar estas operaciones es necesario definir una medida de distancia entre los vectores característicos. Algunas de las medidas de distancia más utilizadas son las distancias métricas inducidas por las normas en espacios Lp.

En el algoritmo de reconocimiento en MATLAB se utiliza una distancia Euclidea, definida del siguiente modo: por ejemplo si $f_i \; y \; f_i^{'} con \; i$ = 0,1,2, … , D son las componentes de dos vectores característicos *f* y *f'*, puede definirse la siguiente métrica inducida por la norma Lp:

$$d = \sqrt{\sum_{i=1}^{D} |f_i - f_i'|^2}$$

Ec 2.1 2 Medida de la Distancia.

En el algoritmo primero se define el tamaño del mayor vector, y se calcula con la fórmula (Ec. 2.12) la distancia entre el vector de la palabra a reconocer y cada uno de los vectores de referencia, que se encuentran en la base de datos, luego se aplican las condiciones para obtener la menor distancia, con lo cual se encuentra la palabra identificada en la base de datos.

4.1.1.6.2. Cálculo de centroides.

Las técnicas de parametrización de la señal de voz se realiza tomando una secuencia de ventanas temporales, cada una de las cuales se representa por un número de *D* parámetros. La información de cada ventana se representa por un vector de observación de *D* posiciones. Cuando se almacenan estos parámetros, se cuantifica cada parámetro con un determinado número de bits, este proceso se denomina cuantificación escalar y es una de las formas empleadas para almacenar la información; además, implica la ocurrencia uniforme de las ventanas de información. Una forma más conveniente es realizar una cuantificación vectorial.

Si se compara la información del vector representante con respecto a la forma de onda original de la señal de voz, se concluye que el análisis espectral contiene significativamente menos información.

Por ejemplo, una señal de voz se muestrea a 10Khz y la cuantificación es de 16 bits, se necesita una velocidad de 160000 bps para almacenar las muestras de la señal de voz en el formato original. Si se realiza el análisis en el espectro,

considerando vectores de dimensión n=10 usando 100 vectores de observación por segundo. Si se representa cada parámetro en 16 bits, se requiere aproximadamente 100x10x16 bps o 16000 bps con una reducción de diez veces sobre la señal original.

Las compresiones en ancho de banda y almacenamiento son imprevisibles, se basan en el concepto de la necesidad de la representación única para cada fonema (sonido diferenciable de una lengua, generalmente representado por una letra), esto puede ser posible para reducir la representación espectral original de la señal de voz sacando los vectores de observación desde un pequeño, finito número de vectores espectrales "únicos", donde cada uno corresponde a las unidades básicas de la voz o "fonemas".

De cualquier forma, el concepto de construir un *codebook* de vectores de análisis, "distintos" y "únicos", aunque con más palabras de código que el grupo o set básico de fonemas, sigue siendo una idea atractiva y es el fundamento de un conjunto de técnicas denominadas métodos de cuantificación de vectores.

Basándose en este razonamiento, se necesita un *codebook* con aprox. 1024 vectores espectrales únicos (25 variantes para cada uno de los 36 fonemas básicos).

Si para representar un vector espectral arbitrario se tiene un número de 10 bits, tomando una velocidad de 100 vectores por segundo, se obtiene una velocidad de 1000 bps para representar los vectores espectrales de una señal de voz.

Esta velocidad es aprox. 1/16 de la velocidad necesaria para vectores espectrales continuos.

Por lo tanto la representación cuantificada es eficiente para representar información espectral de la señal de voz.

Principales ventajas

1. Reduce el almacenamiento de la información de análisis.
2. Se reduce el cálculo para determinar distancias entre vectores espectrales. La representación del VQ se limita a una tabla que contiene las distancias entre pares de vectores del *codebook*.
3. Representación discreta de las señales de voz. Asociando una característica fonética con cada vector del *codebook*, el proceso de elección del vector que mejor lo representa es equivalente a asignar una característica fonética a cada segmento de voz.

Principales desventajas

1. Distorsión en la representación del vector. Hay un número finito de vectores en el *codebook*, el proceso de "elección" del mejor representante es equivalente a cuantificar el vector y conduce a un cierto nivel de error de cuantificación. De cualquier modo con cualquier *codebook* finito siempre habrá un nivel de ruido o error.

2. El almacenamiento requerido para los vectores del *codebook* no es pequeña. Cuanto más grande sea el *codebook* menor es el error. Para un *codebook* de 1000 o más entradas, el almacenamiento no es irrelevante. Hay

que realizar un balance entre error de cuantificación, procesamiento y almacenamiento del *codebook*.

Componentes

Para construir un VQ se necesita:

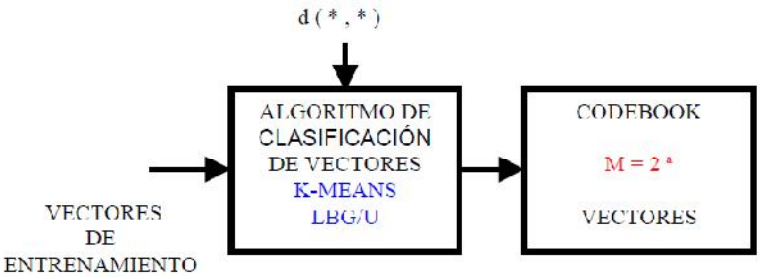

Figura 2. 10 Elementos de un VQ.

1. *Un gran número de vectores de observación*, V1, V2,....., Vn, que conforman el grupo de entrenamiento. El grupo de entrenamiento se usa para crear el grupo de vectores del *codebook* "optimo" que representa la variabilidad espectral observada en el grupo de entrenamiento. Se determina el tamaño del *codebook* como $M = 2^a$, siendo *a* el número de bits necesarios para codificar *M* palabras de código, *por* lo tanto se necesitan $n>>M$ vectores para que sea eficaz. De acuerdo a las pruebas que se realizaron en el transcurso del desarrollo del proyecto se determino que el tamaño apropiado para el codebook es de 64 bits. Ya por medio de pruebas experimentales se comprobó que es la mejor opción puesto que los valores destinados para los centro no solapan (Anexo D).

2. *Una medición de distancia* entre cada par de vectores espectrales de observación para agrupar el conjunto de vectores de entrenamiento como así también para asociar o clasificar vectores arbitrarios a cada entrada del *codebook*.
3. *Un procedimiento de clasificación para ubicar y calcular los centroides*. Sobre la base del particionamiento que clasifica el grupo de n vectores en *M* clusters o sectores primero se elige el número *M, codewords* del *codebook,* para luego proceder a la clasificación.
4. *Finalmente,* luego del proceso de clasificación (entrenamiento) queda como resultado del mismo un libro de códigos o *codebook.*

Grupo de entrenamiento del VQ para reconocimiento de voz

Para entrenar apropiadamente el *codebook* y mejorar la implementación, para el grupo de vectores de entrenamiento, se deberá tener en cuenta:

1. Para las señales de voz:
   - Rangos de edad, acentuación, género, velocidad de discurso, niveles y otras variables.
2. Condiciones de discurso:
   - Ambiente ruidoso o silencioso, movilidad de la persona.
3. Transductores y sistemas de transmisión:
   - Ancho de banda del micrófono, canal telefónico, ancho de banda del canal y otros dispositivos.

4. Reconocimiento discreto o de palabras aisladas y reconocimiento continúo.

Clasificación de Vectores

El objetivo de un módulo clasificador es agrupar una cantidad de vectores característicos, ***N***, en una cantidad *M (M<N)*, discreta, de sectores o celdas de clasificación logrando que las características en cada sector sean similares.

Existen muchos criterios para lograr dicho objetivo y a continuación se verá algunos de los más comunes.

Imagínese que la media multidimensional de un determinado sector *i*, es *μ*i (con 1<*i*<*M*), y a continuación ingresa al clasificador un vector de observación *o*, se puede clasificar dicho vector calculando la "distancia" a la que se halla de cada una de las *M* medias y asignándolo al sector más "cercano".

Este método de clasificación se denomina *k-Means* debido a que se agrupan los vectores en torno a *k* valores medios, quedando formados *k* sectores (en nuestro caso *k*=*M*). Existe el problema de inicialización de los valores de *μi*, y su reestimación a medida que progresa el algoritmo.

Algoritmos de Clasificación

Se puede decir, en general, que los *N* vectores originales de tamaño *D* quedarán representados por *M* vectores, cada uno de los cuales es llamado "palabra de código" o *codeword* (*Cw*), el grupo entero de dichos vectores, forma un "libro de códigos" o

codebook, quedando delimitadas *M* regiones o sectores, llamados regiones de *Voronoi*, determinados por la siguiente expresión:

$$V_i = \{x \epsilon \Re^D : d(x, Cw_j)\} \forall\, j \neq i \; con \; 1 \leq i \leq M$$

Ec 2.1 3 Regiones de Voronoi.

En la figura 2.11 se observa un diagrama de *Voronoi* con sus correspondientes sectores, conformando un *codebook* con sus correspondientes *codewords* como centroides.

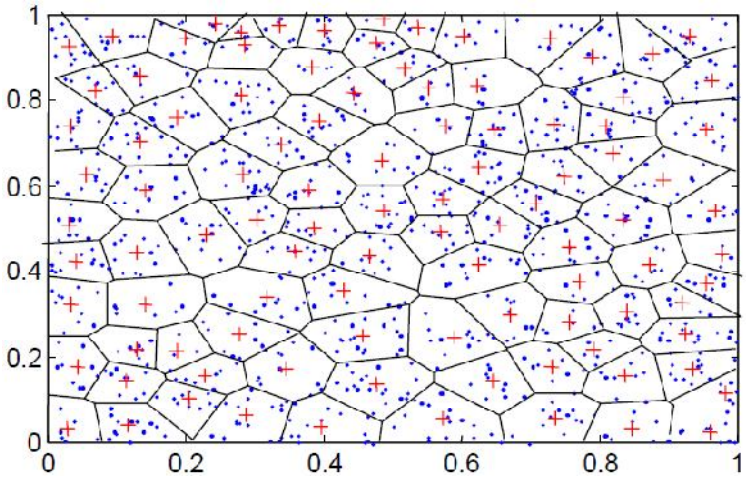

Figura 2. 11 Diagrama de Voronoi

Un procedimiento de cuantificación para señal de voz elige el vector más cercano del *codebook* al vector de observación y utiliza ese vector denominado *codeword,* como la representación resultante para etapas posteriores. Se refiere como al vector "vecino" más cercano, toma como entrada, vectores de señal de voz y da como respuesta, a su salida, el vector que mejor representa esa entrada.

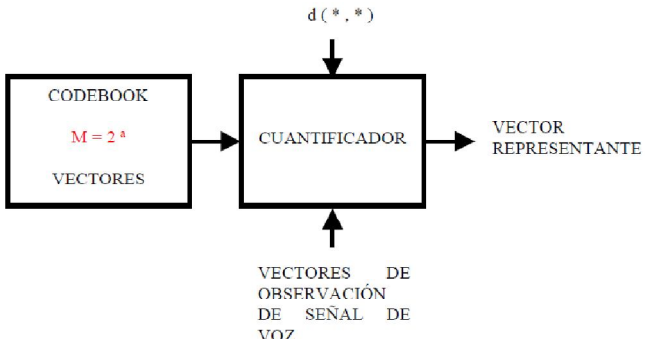

Figura 2. 12 Procedimiento de cuantificación para la señal de voz.

Aplicaciones en Reconocimiento de Voz

1. El uso de múltiples *codebooks* en los cuales cada *codebook* se crea separadamente (e independientemente) para cada una de las representaciones de la señal de voz (espectral o temporal). Por ejemplo se podría crear un *codebook* para las representaciones de los parámetros del *cepstrum* y otro en forma separada conteniendo las representaciones de las derivadas del *cepstrum*.
2. K-tuples cuantificadores en el cual K-tramas de señal de voz se codifican a la vez, en lugar de una única trama como es común. La idea es utilizar las correlaciones en el tiempo entre los sonidos vocales puros y los que tienen componente vocal. La desventaja ocurre cuando los sonidos donde la correlación a través del cuantificador es baja, como son los sonidos transitorios y consonantes.
3. Cuantificación de matrices en las cuales se crea un *codebook* de sonidos o palabras de secuencia de longitud variable. El concepto es manejar la variación temporal vía algunos tipos de procedimientos dinámicos y

posteriormente crear un *codebook* de secuencias de vectores que representan sonidos típicos.

4. Modelos ocultos de Markov en los cuales ambas reducciones, en tiempo y espectro, se usan para cuantificar la emisión completa de la voz de una manera definida y eficiente.

4.1.1.6.3. Cálculo de mínimos y máximos.

- Una función $y = f(x)$ alcanza un MÁXIMO en x_o cuando existe un entorno de x_o en el que $f(x) \leq f(x_o)$
- Análogamente se dice que alcanza un MÍNIMO en x_o cuando existe un entorno de x_o en el que $f(x) \geq f(x_o)$

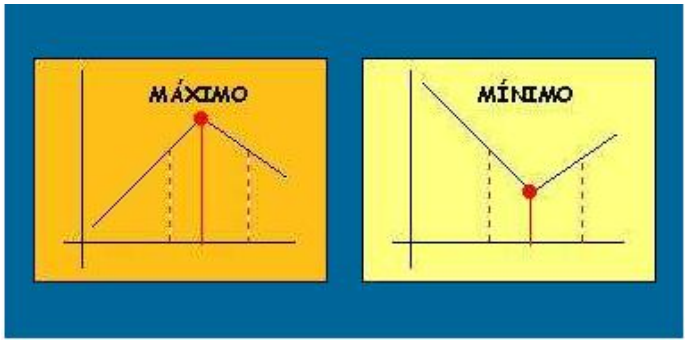

Figura 2. 13 Representación de un mínimo y un máximo.

- **Cálculo de la distancia mínima. Clasificación por medio del Vecino Más Cercano.**

  Se dice que el vecino más cercano de un punto A dentro de un conjunto de puntos S será B si y solo si la distancia considerada entre dichos puntos es la mínima distancia existente entre el punto A y cualquier otro punto dentro del conjunto S:

$$dist(A, B) = min\ dist(A, C). C \in \{S - A\}$$

Ec 2.1 4 Cálculo de distancia mínima.

Cabe señalar el hecho de que si un punto B es el vecino más cercano a otro punto A, no necesariamente habrá de ser A el vecino más cercano al punto B. Cuando un par de puntos A, B cumplen que A es el vecino más cercano a B y a su vez B es el vecino más cercano a A se dirá que el par (A, B) es un par recíproco.

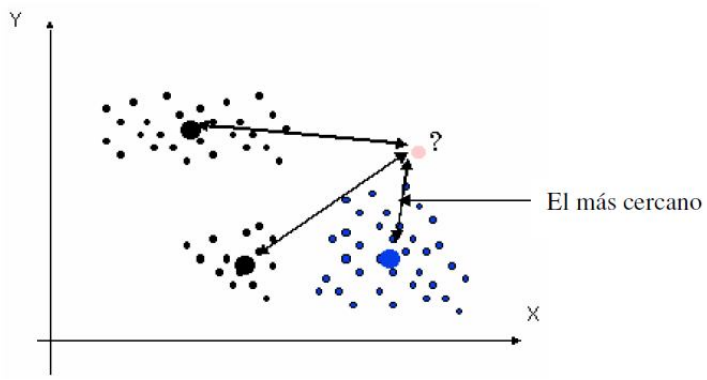

Figura 2. 14 Representación del vecino más cercano

Otra propiedad de esta clasificación es que un punto podrá ser el vecino más cercano de varios puntos dentro de un conjunto por lo que no se podrá entender esta relación como una función.

La técnica del vecino más cercano: esta técnica combina o integra aquellos que tiene la distancia más pequeña entre sí; entonces se computa la distancia entre el grupo recién formado y cada uno de los otros casos fuera de grupo como la distancia mínima entre un caso individual y un caso

perteneciente al grupo. Las distancias entre los casos que no pertenecen al grupo se mantienen sin alterar. En cada paso, la distancia entre dos grupos se define como la distancia que existe entre sus puntos más cercanos.

- **Cálculo de la distancia Máxima**

La técnica del vecino más lejano: la distancia entre dos grupos se define como la distancia entre sus dos puntos más lejanos.

El método de vinculación por el vecino más lejano, el método de vinculación completa o enlace completo y la distancia entre dos conglomerados, se calcula como la distancia entre sus dos elementos más alejados, es decir, la distancia entre dos conglomerados A y B se calcula como

$$dAB = max(d_{ij})''$$

Ec 2.1 5 Calculo de la distancia máxima.

4.1.1.6.4. Cálculo de la moda.

La moda es la medida que se relaciona con la frecuencia con que se presenta el dato o los datos con mayor incidencia, con lo que se considera la posibilidad de que exista más de una moda para un conjunto de datos. La notación más frecuente es la siguiente: *Mo* y $\hat{x}$. Esta medida se puede aparecer tanto para datos cualitativos como cuantitativos. Se dice que cuando un conjunto de datos tiene una moda la muestra es unimodal, cuando tiene dos modas bimodal, cuando la muestra contiene

más de un dato repetido se dice que es multimodal y un último caso es cuando ningún dato tiene una frecuencia, en dicho caso se dice que la muestra es amodal.

Moda para datos agrupados

Para determinar la moda de datos agrupados en clases de igual tamaño su cálculo se puede realizar de la siguiente forma**:**

$$Mo = L_i + \frac{\Delta f_i}{\Delta f_i + \Delta f_s} A$$

Ec 2.1 6 Moda de datos agrupado.

Donde

$L_i =$ límite inferior o frontera inferior.
$\Delta f_i =$ Exceso de la frecuencia modal sobre la clase modal inferior inmediata.
$\Delta f_s =$ Exceso de la frecuencia modal sobre la clase modal superior inmediata
$A =$ Anchura o intervalo de la clase modal.

En ocasiones la expresión para el cálculo de la moda suele presentarse de la siguiente forma**:**

$$Mo = L_i + \frac{f_m - f_{(m-1)}}{2 f_m - f_{(m-1)} - f_{(m+1)}} A$$

Ec 2.1 7 Calculo de la moda

Donde

$f_m = \text{Frecuencia de clase modal}$
$f_{(m-1)} = \text{Frecuencia de clase premodal}$
$f_{(m+1)} = \text{Frecuencia de clase posmodal}$

Aunque la expresión se ve un poco diferente en realidad se trata de una misma ecuación, ya que el exceso de la clase modal inferior se puede determinar cómo:

$$\Delta f_i = f_m - f_{(m-1)}$$

Ec 2.1 8 Clase modal Inferior.

Y el exceso de la clase modal superior se determina como

$$\Delta f_s = f_m - f_{(m-1)}$$

Ec 2.1 9 Clase modal Superior.

Por lo que basta sustituir estos valores en una de ellas para encontrar la otra expresión.

4.1.2. BLOQUES UTILIZADOS PARA LA PROGRAMACIÓN DE LA TARJETA CON SIMULINK.

Para la programación de la tarjeta con Simulink se debe seguir el procedimiento que se detalla a continuación:

1. Se procederá a abrir un nuevo modelo en Simulink.
2. Seleccionar la tarjeta con la que se va a trabajar.

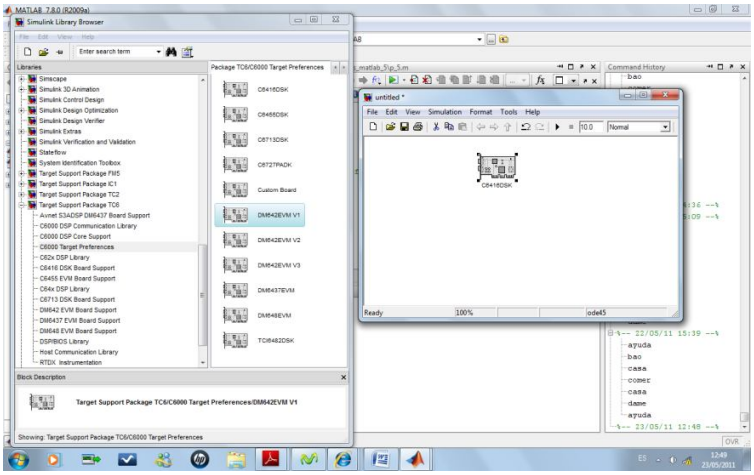

Figura 2. 15 Selección bloque C6416DSK

3. Seleccionar el bloque de reset de la tarjeta.

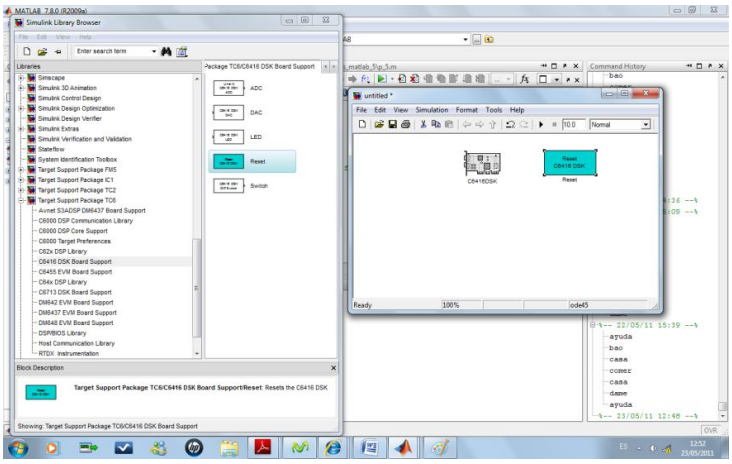

Figura 2. 16 Librería C6416DSK Board Support

4. Seleccione el bloque para la adquisición de los datos a través del puerto Mic In de la tarjeta.

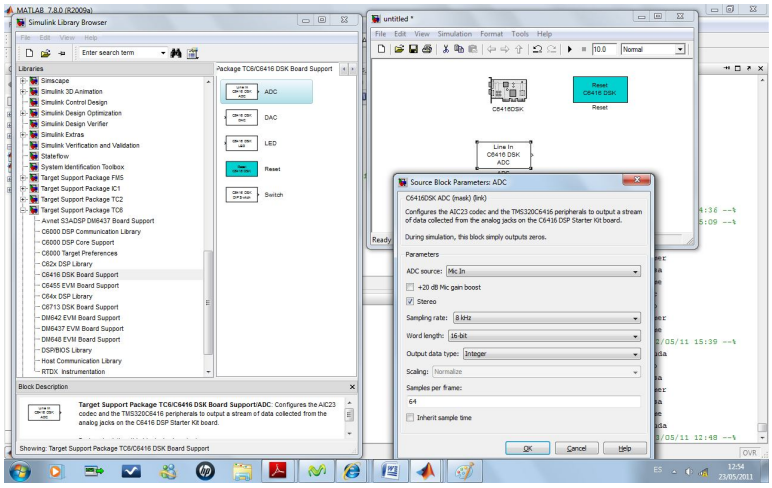

Figura 2. 17 Direccionamiento de la Entrada

5. Seleccione el bloque para la salida de los datos a través del puerto headphone de la tarjeta.

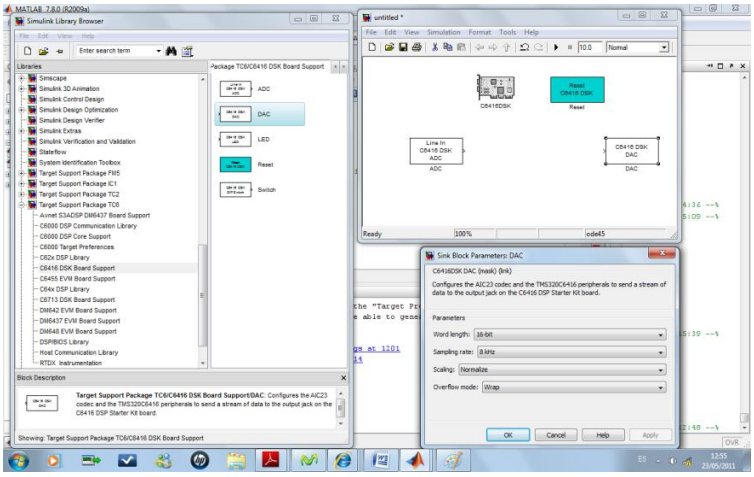

Figura 2. 18 Direccionamiento de la Salida

6. Seleccione el bloque de Embedded MATLAB Function, y asignar el código correspondiente a la función que será la encargada del procesamiento de la señal.

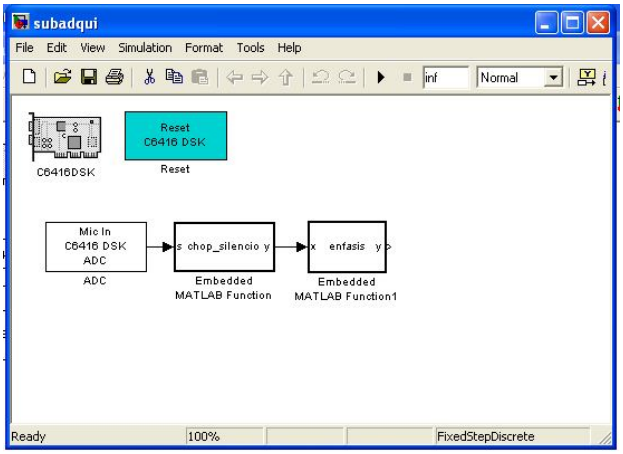

Figura 2. 19 Bloques de Programación.

7. Configurar los parámetros que se muestran a continuación para la programación de la tarjeta. Seleccionar en la barra de menú Simulación/Configuración de parámetros.

   7.1 Seleccionar el tiempo con el que se va a trabajar en nuestro caso infinito ya que siempre se va a tomar muestras.

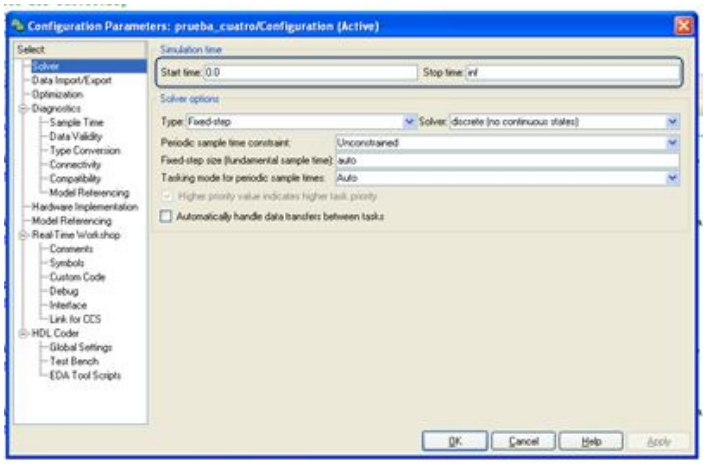

Figura 2. 20 Selección de Tiempo de Muestreo.

   7.2 Seleccionar la tarjeta embebida con la que se va a trabajar. En nuestro caso se seleccionara la TIC6000.

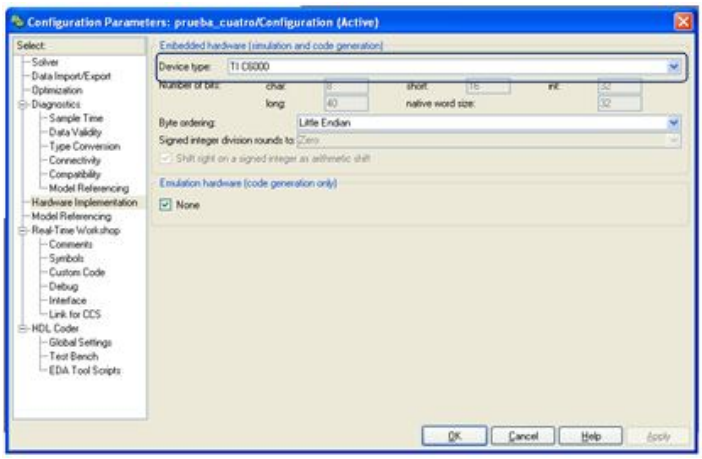

Figura 2. 21 Selección del tipo de dispositivo.

7.3 Seleccionar el archivo con el cual el programa Simulink se va a comunicar con el Code composer studio para la programación de la tarjeta.

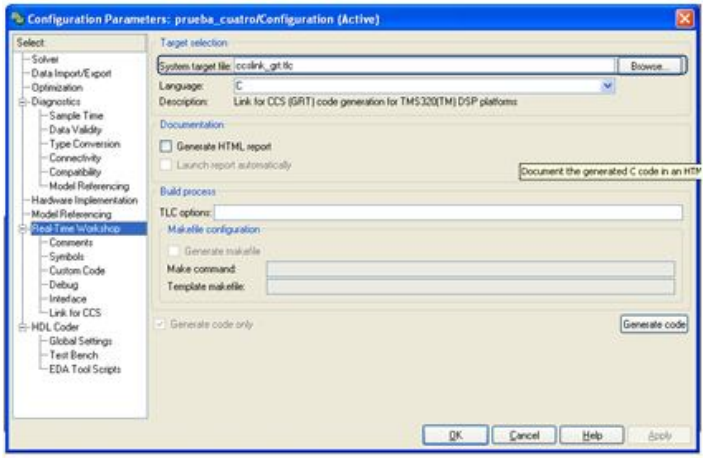

Figura 2. 22 Archivo del sistema de la tarjeta.

7.4 Verificar que en System stack size este 8192 y en runtime este Buil_and_execute.

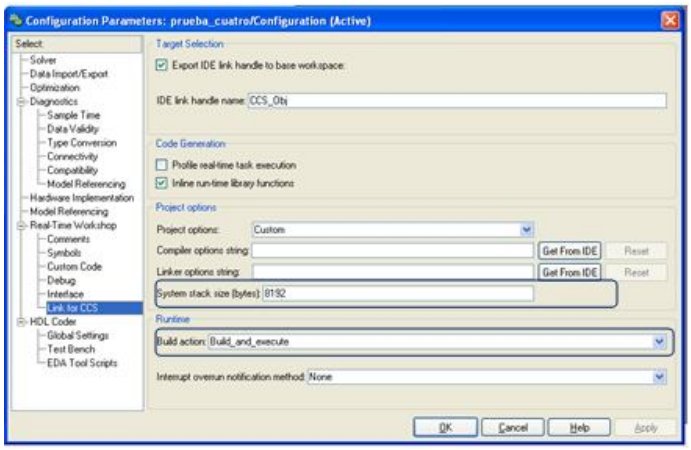

Figura 2. 23 Paleta de configuración para la compilación.

8. Seleccionar en la barra de menú Tools/Real time Workshop/Build Model

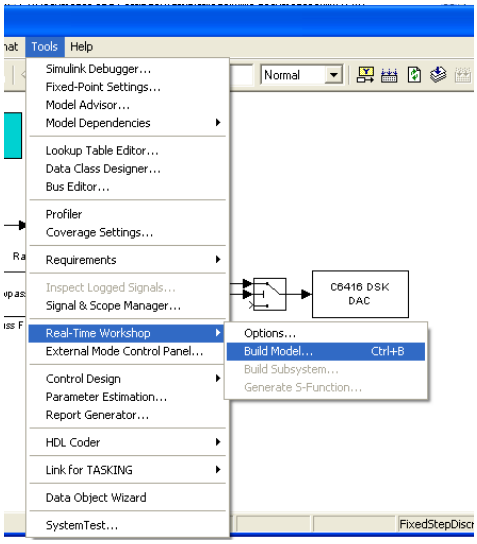

Figura 2. 24 Compilación del programa.

9. Verificar que el programa se cargue correctamente.

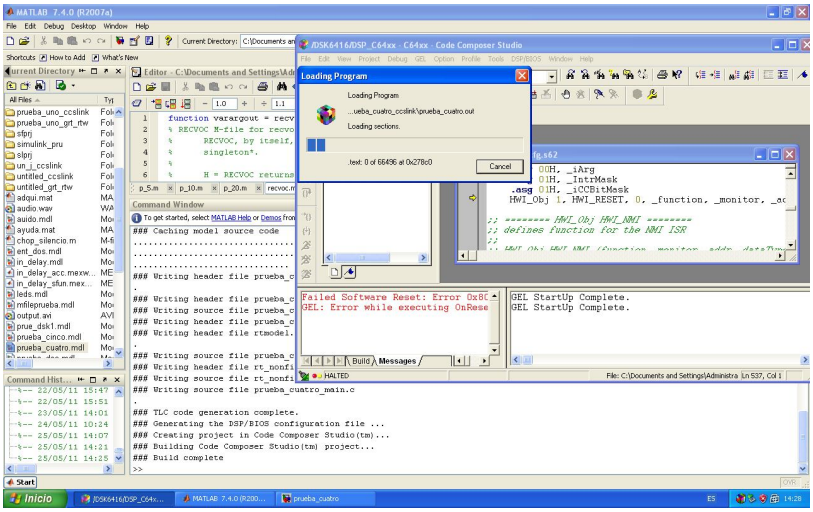

Figura 2. 25 Interacción entre MatLab y Code Composer Studio.

CAPITULO III

PRUEBAS EXPERIMENTALES

3.1 PRUEBAS CON EL SOFTWARE.

El software consta de una interfaz grafica, muy fácil de utilizar, con una base de datos,| con palabras pre-establecidas.

La interfaz grafica se realizo en GUI y se muestra en la figura.

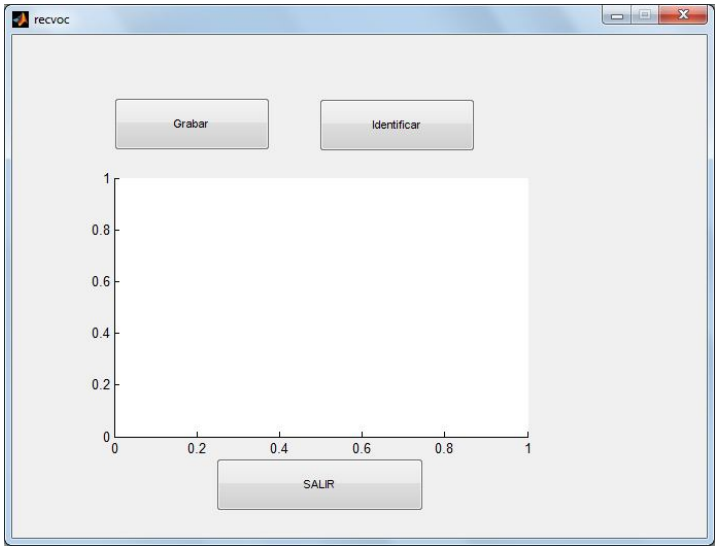

Figura 3. 1 Interfaz de manejo del software

3.1.1 MANEJO DE LA INTERFAZ DE RECONOCIMIENTO DE VOZ.

Para evaluar el sistema, se obtiene una señal de muestra y se sacan las características de esta señal para ser comparada con cada una de las características almacenadas en la base de datos. Para esta comparación se utiliza una medida de distancia Euclidiana.

1. Presionar el botón de grabar, y pronunciar la palabra a reconocer.

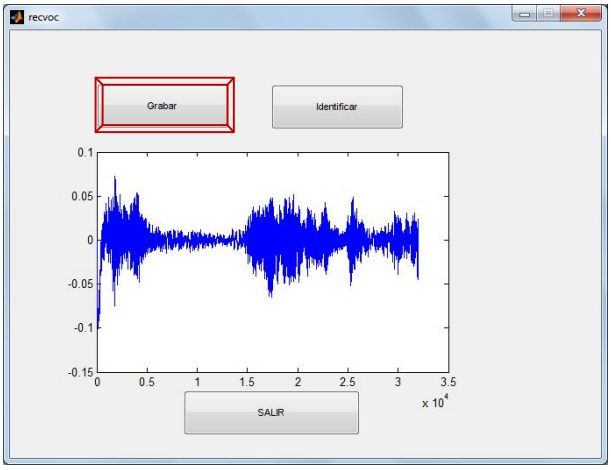

Figura 3. 2 Grabación de la palabra

2. Presionar el botón identificar, el cual da como resultado la palabra identificada, ver figura 3.3

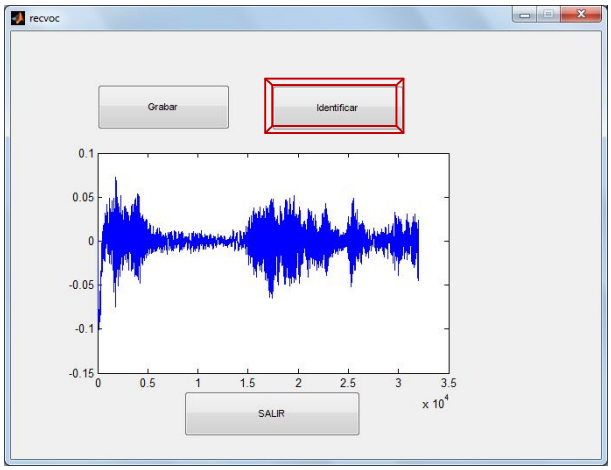

Figura 3. 3 Identificación de la palabra

3. Palabra identificada mostrada de manera grafica. Además se escucha la palabra con voz clara.

Figura 3. 4 Palabra Reconocida.

3.2 DESCRIPCIÓN FÍSICA DEL DISPOSITIVO.

Se presenta el dispositivo reconocedor de palabras utilizando algoritmos para los coeficientes Cepstrales y las medidas de distancia, cálculo de moda y media; se describe a continuación sus componentes.

Figura 3. 5 Dispositivo de reconocimiento de voz para personas con discapacidad en el habla.

El dispositivo de reconocimiento de voz para personas con discapacidad en el habla consta de las siguientes partes:

- **Alimentación.** El dispositivo se alimenta con 5 voltios y 3 Amperios. En la tarjeta existe un regulador de voltaje que

proporciona 1.2 V para el núcleo del DSP y 3.3 V para las entradas y salidas.

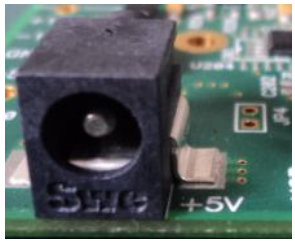

Figura 3. 6 Puerto de alimentación del dispositivo.

- **Puerto para la programación del dispositivo.**

  A través de este puerto USB conector (J201), se procederá a cargar el programa de reconocimiento de voz desde el computador hasta el DSP.

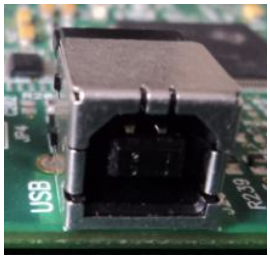

Figura 3. 7 Puerto de comunicación entre el dispositivo y el computador.

- **Entrada.**

  A través de este puerto se realiza la adquisición de los datos a través del micrófono laringófono. El puerto utilizado es el Mic In.

Figura 3. 8 Entrada de audio Mic In del Dispositivo.

- **Salida.**

  La salida que produce el dispositivo se la realiza a través del puerto Headphone del dispositivo de reconocimiento de voz.

Figura 3. 9 Salida del dispositivo de reconocimiento de voz.

3.2 RESULTADOS OBTENIDOS.

a. Pruebas con la Señora María.

Figura 3. 10 Dispositivo conectado a la señora María

Para las pruebas con la señora María se le ha pedido que pronuncie las siguientes palabras.

- ✓ Ayuda
- ✓ Baño
- ✓ Casa
- ✓ Comer
- ✓ Dame

De las cuales se obtuvieron los siguientes resultados.

Tabla 3.1: Porcentaje de reconocimiento colaboradora María.

PALABRA	NUMERO DE REPETICIONES					PORCENTAJE POR PALABRA
	1	2	3	4	5	
AYUDA	X	X	X	X	X	100%
BAÑO	X	-	X	X	X	80%
CASA	X	X	X	X	X	100%
COMER	X	X	X	-	X	80%
DAME	-	X	X	X	X	80%
PORCENTAJE POR BASE DE DATOS	80%	80%	100%	80%	100%	
PORCENTAJE TOTAL DE RECONOCIMIENTO DE CADA PALABRA						88%
PORCENTAJE TOTAL DE RECONOCIMIENTO DE LA BASE DE DATOS						88%

De acuerdo a la tabla mostrada en la parte superior se ha concluido que el porcentaje de reconocimiento de voz en la señora María es del 88%.

b. Pruebas con el señor Juan.

Figura 3. 11 Dispositivo conectado al señor Juan.

Para las pruebas con el señor Juan se le ha pedido que pronuncie las siguientes palabras.

- ✓ Ayuda
- ✓ Baño
- ✓ Casa

- ✓ Comer
- ✓ Dame

De las cuales se obtuvieron los siguientes resultados.

Tabla 3.2: Porcentaje de reconocimiento colaborador Juan.

PALABRA	NUMERO DE REPETICIONES					PORCENTAJE POR PALABRA
	1	2	3	4	5	
AYUDA	X	X	X	-	X	80%
BAÑO	X	-	X	X	-	60%
CASA	X	X	X	X	X	100%
COMER	X	X	-	X	X	80%
DAME	-	-	X	X	X	60%
PORCENTAJE POR BASE DE DATOS	80%	60%	80%	80%	80%	
PORCENTAJE TOTAL DE RECONOCIMIENTO DE CADA PALABRA						76%
PORCENTAJE TOTAL DE RECONOCIMIENTO DE LA BASE DE DATOS						76%

De acuerdo a la tabla mostrada en la parte superior se ha concluido que el porcentaje de reconocimiento de voz en el señor Juan es del 76%. Presentándose un menor porcentaje de reconocimiento en las palabras baño y dame.

c. Pruebas con el señor Fernando.

Figura 3. 12 Dispositivo conectado al señor Fernando

Para las pruebas con el señor Fernando se le ha pedido que pronuncie las siguientes palabras.

- ✓ Ayuda
- ✓ Baño
- ✓ Casa
- ✓ Comer
- ✓ Dame

De las cuales se obtuvieron los siguientes resultados.

Tabla 3.3: Porcentaje de reconocimiento colaborador Fernando.

PALABRA	NUMERO DE REPETICIONES					PORCENTAJE POR PALABRA
	1	2	3	4	5	
AYUDA	X	-	X	X	-	60%
BAÑO	-	X	-	X	X	60%
CASA	X	X	-	-	X	60%
COMER	-	-	X	-	-	20%
DAME	-	-	X	X	X	60%
PORCENTAJE POR BASE DE DATOS	40%	40%	60%	60%	60%	
PORCENTAJE TOTAL DE RECONOCIMIENTO DE CADA PALABRA						52%
PORCENTAJE TOTAL DE RECONOCIMIENTO DE LA BASE DE DATOS						52%

De acuerdo a la tabla mostrada en la parte superior se ha concluido que el porcentaje de reconocimiento de voz en el señor Fernando es del 52%. Presentándose un menor porcentaje de reconocimiento en la palabra comer. En este caso se ha disminuido el porcentaje de reconocimiento en general, debido a que el señor Fernando tiene un alto grado de discapacidad auditiva.

3.4 ANÁLISIS DE COSTOS

El presente proyecto tiene un costo de $755.00, en los cuales no se incluyen los gastos de mano de obra.

El costo de cada uno de los elementos utilizados en el desarrollo del proyecto se especifica a continuación:

Tabla 3.3: Detalle de elementos y costos.

DETALLE	UNIDAD	PRECIO ($)
Micrófono Laringófono	1	45.00
TMS320C6416	1	670.00
Parlantes	1	20.00
Otros	1	20.00
TOTAL		755.00

El costo de la tarjeta es elevado, ya que la misma es de propósito general, se puede disminuir el costo adquiriendo una tarjeta de propósito específico, dedicada al procesamiento de señales de voz.

3.5 ALCANCES Y LIMITACIONES.

ALCANCES

Este proyecto se desarrolló para reconocer veinte palabras, que puedan emplear personas discapacitadas, y así lograr mejorar la comunicación con las personas que interactúa a diario.

Las palabras que reconoce este dispositivo, son las de mayor frecuencia de uso, se determino en base a los requerimientos de la persona, a las cuales se les hizo la prueba.

El espacio de memoria que ocupa el programa en el dispositivo es de aproximadamente 500Kbytes de memoria, lo que posibilita realizar un reconocimiento óptimo de hasta 5 palabras.

LIMITACIONES

- ✓ El proyecto está limitado para las personas con discapacidad física en el habla; este dispositivo permite reconocer la voz de aquellos individuos, cuyas cuerdas vocales se encuentren sanas, mientras su caja fonadora pueda presentar algunas patologías o inconvenientes por algún tipo de mal formación o enfermedad, que impida el reconocimiento de las palabras de manera convencional.
- ✓ Debido a que los colaboradores en las pruebas de este proyecto, debían tener ciertas características de discapacidad, se debe pedir que un especialista en Otorrinolaringología, verifique que las cuerdas vocales funcionan normalmente.
- ✓ Los colaboradores con discapacidad mental, no son elementos adecuados para el uso de este dispositivo.
- ✓ El espacio de memoria requerido para este proyecto es alto para un DSP, por lo que la programación está restringida a cinco palabras, sin embargo, se puede realizar la demostración empleando un PC, cuya capacidad, permitiría cubrir mayor número de palabras.
- ✓ Se debe adecuar las condiciones de almacenamiento de información para cada persona discapacitada; debido a que las frecuencias son diferentes; por lo cual el número de colaboradores también fue reducido a tres participantes.
- ✓ El dispositivo está limitado para el uso de una sola persona, por el empleo de la base de datos que tiene ya palabra preestablecida para un usuario, para otro usuario se deberá restear el chip.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Se desarrolló un dispositivo de reconocimiento de voz para personas con discapacidad en el habla, con limitaciones en el número de palabras, y con un porcentaje de confiabilidad promedio del 72%, dependiendo esta cantidad de la discapacidad de los colaboradores.
- Se verificó que con la herramienta MATLAB se reduce la complejidad del procesamiento digital de señales, por su gran capacidad de procesamiento en tiempo real, y en uso de diferentes toolboxs que permite el manejo de ecuaciones complejas.
- Mediante el procesamiento de señal se comprobó que, el espectro de la señal brinda información relevante de las señales de voz.
- Para identificar mejor la señal es necesario depurarla con el uso de filtros para enfatizar las características acústicas.
- De los métodos de procesamiento para reconocimiento de voz estudiados en el transcurso de la elaboración del proyecto se determino que, la extracción de las características de la voz por medio de los LPC cepstrum fue el de mayor precisión.
- Se observó que el principio básico de la cuantificación vectorial es comprimir la cantidad de datos a procesar por medio del uso de un codebook con un número relativamente pequeño de centroides.

- Se determinó en forma experimental la centroide de cada palabra hablada, la cual adquiere una característica fonética diferente, clasificando estas según la información espectral que representan.
- Se llegó a un compromiso entre error por distorsión, almacenamiento y costo computacional; lo que produce un adecuado nivel de satisfacción, que corresponde a un alto porcentaje de reconocimiento de voz en discapacitados físicos.
- Se comprobó que los parámetros LPC Cepstrum, funcionan de manera excelente con un número reducido de coeficientes, típicamente 10, dependiendo de la frecuencia de muestreo, y se obtiene un moldeamiento eficiente de las características del tracto vocal.
- Se evidenció que los coeficientes LPC Cepstrum, requieren un costo computacional reducido, característica primordial a tener en cuenta en el momento de la aplicación, porque el dispositivo requiere un compromiso entre eficiencia y velocidad de procesamiento para su correcto funcionamiento.
- Al emplear el algoritmo de las medidas de distancia euclidiana y el uso del cálculo del vecino más cercano y lejano, en la aplicación del reconocimiento de voz, los resultados de identificación asciende hasta un 80% de exactitud.
- Se logró un 80% de exactitud por palabra, que según la investigación bibliográfica recopilada durante este proyecto; se considera lo suficientemente alto, debido a la complejidad de las características que presenta la voz y el grado de incertidumbre en identificar la voz de las personas con

discapacidad en el habla, porque la voz es una señal transitoria.

- El DSP utilizado para este proyecto es óptimo para el trabajo, con señales digitales, consistente en cálculos de señales de audio.
- La tarjeta TS, permite programar el DSP, por tanto puede emplearse en otros tipos de aplicaciones.
- En las pruebas realizadas con el tercer colaborador se obtuvo un reconocimiento del 52% ya que esta persona presenta un mayor grado de dificultad auditiva en comparación con las dos primeras personas.
- Los errores de compilación desde Matlab-Simulink debido a la selección del compilador por defecto, fue superado cuando se eligió el ccslink adecuado.
- Se estableció que es relevante que se encuentre instalado Visual Studio C/C++ 5, 6,7; conteniendo estos los archivos DLL necesarios para realizar la compilación del programa.

RECOMENDACIONES

- Verificar que el MATLAB que se esté utilizando posea todas las librerías para el manejo de tarjetas embebidas.
- Leer las guías de ayuda de Matlab para resolver los problemas que se presentan en el momento de la compilación.
- Verificar que la tarjeta este conectada al momento de compilar el programa para evitar errores futuros.
- Se puede investigar y aplicar otros métodos de reconocimiento diferentes al utilizado en este trabajo como fueron: los modelos ocultos de Markov, Redes neuronales, alineamiento temporal

dinámico; para hacer un comparativo del porcentaje de reconocimiento, entre varias opciones.

- Para optimizar el funcionamiento del dispositivo de reconocimiento, se recomienda implementar un sistema de reconocimiento de palabras para un discurso continuo.
- Se puede utilizar otros tipos de DSP, que dispongan de mayor capacidad de memoria como el TMS320C6173; lo que permitiría, disponer de una mayor base de datos para el reconocimiento, y mejorar la tecnología que existe en los laboratorios del Departamento de Eléctrica y Electrónica.

BIBLIOGRAFÍA Y ENLACES

- ✓ http://es.wikipedia.org/wiki/Conversor_anal%C3%B3gico-digital
- ✓ http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/de_l_g/capitulo2.pdf
- ✓ http://www.google.com.ec/url?sa=t&source=web&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fwww.unav.es%2Frevistamedicina%2F50_3%2F1.HISTORIA%2520DE%2520LA%2520VOZ.pdf&ei=3H2vTPWeCcKAlAfT35nlDw&usg=AFQjCNEWQaIuWTnF8Dfsei7a3q9XV6S_IA
- ✓ http://www.fundacionbelen.org/base_datos/habla.htm
- ✓ http://www.encolombia.com/medicina/otorrino/otorrinosupl31203-contenido.htm
- ✓ http://www.secyt.frba.utn.edu.ar/gia/IA1_IntroReconocimientoVoz.pdf
- ✓ http://www.secyt.frba.utn/INSN0506-TemaProduccionDeHabla-JMG-v46 .pdf
- ✓ http://mayramartinezplana.net/files/11086-10660/Habla_y_lenguaje.pdf
- ✓ http://www.gcnlevante.com/radiocom/uso_libre/motorola/accesorio_motorola__laringofono_x20m.htm#
- ✓ http://edwsoft-dspti.blogspot.com/2009/05/dsp-tms320c6416-dsk-de-texas.html
- ✓ http://www.usc.es/gir/docencia_files/tdd/tutorial_matlab.pdf

- ✓ http://www.google.com/url?sa=t&source=web&cd=2&ved=0CBgQFjAB&url=http%3A%2F%2Fwww.gnewbook.org%2Faction%2Ffile%2Fdownload%3Ffile_guid%3D70523&rct=j&q=DISE%C3%91O%20E%20IMPLEMENTACI%C3%93N%20DE%20UN%20PROTOTIPO%20DE%20RECONOCIMIENTO%20DE%20VOZ%20BASADO%20EN%20MODELOS%20OCULTOS%20DE%20MARKOV%20PARA%20COMANDAR%20una%20silla%20de%20ruedas%2Bpdf&ei=7QiITfnfKMux0QHvhsHfDQ&usg=AFQjCNEkXdh0tg5rrB2jhYHt0QBVnZ3aZw

- ✓ RULPH CHASSAING, Digital Signal Processing and Applications with the C6713 and C6416 DSK

- ✓ KALURI V. RANGARAO, Digital Signal Processing A Practitioner's Approach.

ANEXOS.

ANEXO A

TMS320C6416 DSK.

El TMS320C6416 DSK, nos ayuda con la evaluación y desarrollo de aplicaciones, como esquemas lógicos, ecuaciones lógicas. Dispone de un hardware que reduce el tiempo de desarrollo en las aplicaciones.

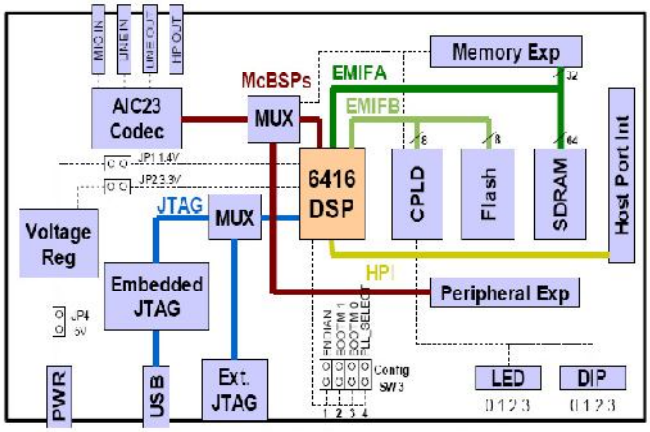

Figura A.1 Diagrama de Bloques del DSK C6416.

Dentro de las características podemos citar las siguientes:

- El Procesador Texas Instruments TMS320C6416 DSP, opera de 600 a 720 MHz
- Posee en tarjeta un AIC23 Codificador Stereo.
- Una DRAM sincrónica de 16MB.
- 512 KB de memoria FLASH no volátil.
- LEDS y DIP conmutadores, para el usuario.
- Software de configuración a través de los registros implementados en CPLD.

- Opciones de arranque configurable y selección de entrada del CLOCK.
- Conectores de expansión para el uso de Daughter Card.
- Fuente de voltaje de +5V.

FUNCIONAMIENTO DEL TMS320C6416 DSK.

El DSP en el 6416 DSK, posee comunicación a través uno de los dos buses, el de 64 bit de ancho EMIFA y el de 8 bit de ancho EMIFB. La SDRAM (EMIFA), Flash (EMIFB) y el CPLD (EMIFB) están conectados a estos buses. EMIFA esta también conectada a la tarjeta de expansión. El Codec AIC23 permite que el DSP transmita y reciba señales Analógicas. McBSP1, es usada para el codec control interfaces. McBSP2, es usada para datos.

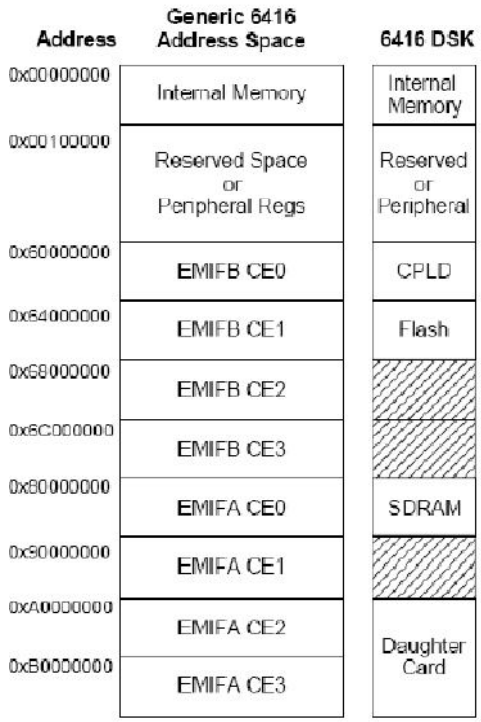

Figura A.2 Mapeo de Memoria del C6416 DSK.

Las entradas y salidas analógicas están localizadas a través de cuatro conectores de audio de 3.5 mm que corresponden a:

- MIC IN: Micrófono señal de entrada.
- LINE IN: Línea de Entrada.
- LINE OUT: Línea de Salida, ganancia fija.
- HP OUT: Audífonos de Salida, ganancia ajustable

El CODEC puede seleccionar el micrófono o la línea de ingreso como ingresos activos. McBSP1 y McBSP2 pueden ser re-enrutados a los conectores de expansión a través de software. El CPLD (Dispositivo Lógico

Programable), es usado para implementar glue logic que están en la tarjeta.

El CPLD también tiene una interface de usuario basada en un registro, que permiten al usuario configurar la tarjeta, para que lea o grabe los registros

CPLD.

El DSK incluye cuatro LEDS y cuatro DIP switch de simple vía que permite interactuar al usuario. Ambos son accesos realizados a la lectura y escritura de los registros CPLD. Incluye una fuente de poder externa de +5V, que se usa para energizar la tarjeta.

El voltaje regulado conmutado es de 1.4V para el núcleo del DSP y se suministra 3.3V a las entradas y salidas. Un regulador de voltaje separado energiza con 3.3V a la interface de expansión.

CCS se comunica con el DSK a través de un emulador JTAG embebido con una interface con el USB Host. El DSK también puede ser usado con un emulador externo a través del conector JTAG externo.

OPERACIÓN BÁSICA.

El DSK esta diseñado para trabajar con el CCS de TI con la versión específica para trabajar con la tarjeta. El CCS se comunica con la tarjeta a través del emulador JTAG presente en la tarjeta. Para iniciar, sigua las instrucciones en la guía de instalación (ver Anexo GUIA DE INSTALACION) para instalar el CCS. Este brindara la instalación de las herramientas de desarrollo, documentación y controladores. Después de la instalación completa, debe seguir los siguientes pasos para correr CCS.

1. Conecte la fuente de Poder del DSK.
2. Conecte el DSK a su computadora por medio del cable USB.
3. - Ejecutar CCS dando clic en el icono en su escritorio.

ANEXO B

GUIA DE INSTALACION.

El kit TMS320C6416 DSK contiene:

- Una tarjeta DSK C6416.
- Fuente de poder universal de +5V.
- Cable de poder AC.
- CD-ROM con Code Composer Studio para el DSK C6416.
- Manual de Referencia Tecnica TMS320C6416 DSK.
- Cable USB.

REQUERIMIENTOS DE HARDWARE Y SOFTWARE.

Estos requerimientos son necesarios para instalar el Code Composer Studio IDE y soportar el puerto USB. Los requerimientos para la plataforma son:

CONFIGURACIÓN MÍNIMA DEL HARDWARE.

- Pentium de 233MHz o superior.
- 600 MB de disco duro disponible.
- Windows 98SE, 2000 o XP.
- 64MB de RAM.
- CD-ROM Drive

CONFIGURACIÓN DE HARDWARE RECOMENDADA.

- 128MB de RAM.
- Monitor SVGA (1024x768) color.

- Pentium 500MHz o superior.

El DSK de 720MHz es una versión actualizada del original del DSK de

600MHz, el cual incluye modificaciones de hardware y software.

INSTALACIÓN DEL DSK CODE COMPOSER STUDIO.

Antes de instalar el Software DSK asegúrese que su PC tiene USB y sistema operativo que soporte este puerto.

Para Windows 2000 y XP debemos instalar el CCS en la sesión de administrador. Para ejecutar el CCS en estos sistemas se requiere permiso de escritura en el registro. Si instalas el hardware seguir las instrucciones que vienen con el hardware. También antes de instalar el DSK CCS asegúrese de que el antivirus este deshabilitado. Este puede ser habilitado nuevamente cuando ejecutes el CCS.

1. Insertar en CD con CCS en el CD-ROM drive. Una ventana de instalación debe aparecer, si no ir a Explorador de Windows y ejecutar Setup.exe desde el CDROM.
2. Escoger una de las siguientes opciones de instalación:

- Code Composer Studio. Instalación completa de CCS para el C6416DSK.
- C6416DSK incluido en CCS v2.21. parche para una versión existente de CCS 2.21. Requiere CCS v2.21. Si se selecciona esta opción entonces ejecutar CCS 2 (6000) e importar la configuración “TMS320c6416dsk – 0x540”

3. Responder a los cuadros de dialogo mientras las instalación se ejecuta.

4. Deje el CD de CCS en el CD-ROM, este se usara para instalar el USB hardware. Reinicie el PC.

El DSK CCS v2.21 automáticamente configurara tu sistema con una configuración pre ajustada para el dispositivo C6416 DSK USB.

El proceso de instalación creara dos iconos en el escritorio:

C6416 DSK startup – C6416 DSK CCS

C6416 DSK Diagnostic Utility.

CONECTANDO EL C6416 DSK AL PC:

1. Conecte el cable USB al PC o Laptop. Si conectas el cable USB a un USB HUB, debes estar seguro que el HUB esté conectado a tu PC o Laptop.
2. Si usted planea instalar un micrófono, parlante o tarjeta de expansión, estas deben estar conectadas apropiadamente antes de activar el DSK.
3. Conecte el cable de poder AC a la fuente de poder.
4. Precaución: el cable de poder debe estar conectado a la fuente AC antes de conectar los 5V DC al DSK.
5. Conecte el cable de poder a la tarjeta.
6. Cuando el poder es aplicado a la tarjeta, el Power-ON self test (POST) se ejecutara. Los LEDS de cero a tres parpadearan. Cuando el POST esta completo todos los LEDS parpadearan y luego se quedaran encendidos.
7. En este punto su DSK está funcionando y usted puede ahora terminar la instalación del USB.
8. Asegúrate de que el CCS CD-ROM DRIVE este instalado en su PC.

Ahora conecte el cable USB en el DSK, luego de pocos minutos Windows detectara el nuevo hardware.

Siga las instrucciones en la pantalla y deje que Windows encuentre los controladores del USB sdusbemu.inf y sdusdemu.sys están en su CCS CD-ROM. En Windows XP deberán encontrar los controladores automáticamente.

INICIANDO EL CODE COMPOSER STUDIO.

Para iniciar Code Composer Studio, haga doble clic en el icono de C6416 DSK CCS en escritorio del PC.

CORRIENDO EL TUTORIAL DEL CODE COMPOSER.

La ayuda en línea incluida en el C6416 DSK contiene información fundamental acerca del hardware y el software que contiene el Kit. Esta también contiene el tutorial que ayudara a iniciar su DSK y puede aprender acerca de sus características. Para acceder a la ayuda en línea y correr el tutorial, siga los siguientes pasos:

1. Inicie CCS (ignore este si CCS está corriendo) con un doble clic en el icono de tu escritorio.
2. Abra el CCS seleccionando Help – Contents en el menú del CCS.
3. Abra la ayuda específica del DSK C6416 abriendo el tópico etiquetado TMS320C6416 DSK. Este aparece en la parte última de dicho tópico.
4. Mire en la sección titulada Welcome to Your C6416 DSK, aquí encontrara el tutorial y otros materiales de introducción.

ANEXO C

La aplicación C6416 Diagnostics, realiza un chequeo General y Avanzado del DSK para verificar la posible existencia de problemas de hardware.

El chequeo General de los componentes de la tarjeta verifica:

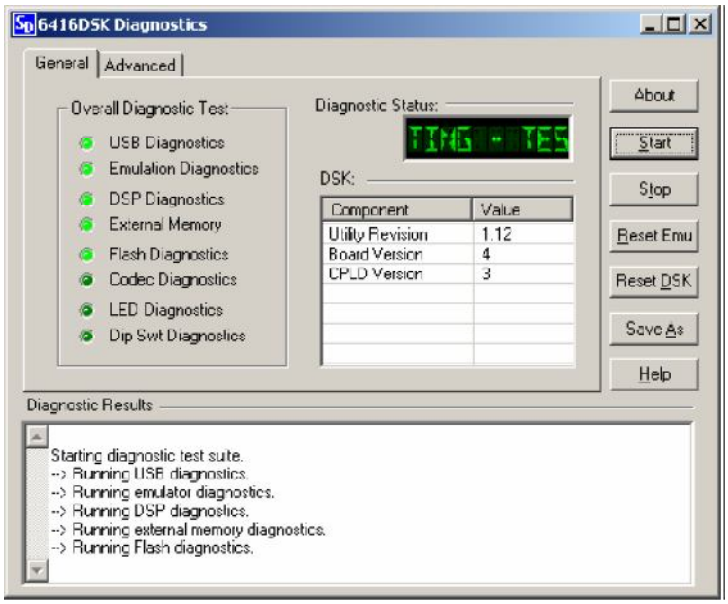

Figura C.1 Diagnostico General.

DIAGNOSTICO USB.- Detecta e inicializa e USB emulador de la tarjeta.

DIAGNOSTICO EMULACION.- Chequea que el emulador se comunica con el JTAG del DSP.

DIAGNOSTICO DSP.- Corre internamente un análisis en el núcleo del DSP y sus periféricos.

MEMORIA EXTERNA.- Ejecuta y direcciona análisis en la SDRAM externa.

DIAGNOSTICO FLASH.- Programa y verifica un patrón en la external Flash.

DIAGNOSTICO CODEC.- Genera un tono al parlante y línea de salida, muestra la línea de ingreso.

DIAGNOSTICO LED.- Corre un destello de los LED de izquierda a derecha que un usuario puede visualizar y verificar.

DIAGNOSTICO DIP SWITCH.- Muestra el valor de los Switch con luz en su correspondiente LED.

El chequeo Avanzado, contiene análisis que permite al usuario chequear l operación de un dispositivo con particular detalle.

Seleccionamos un determinado análisis al hacer clic en un botón en el panel. Los botones de control se encuentran en el lado derecho de la aplicación, tiene la misma función que la función General, pero cada análisis lo realiza de manera individual.

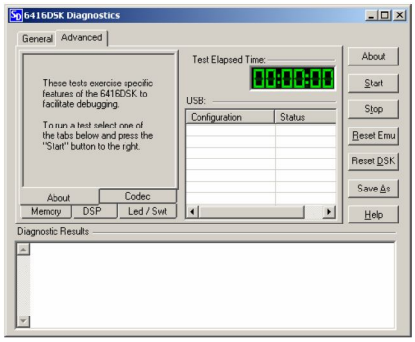

Figura C.2 Diagnostico Avanzado.

Dentro de los análisis que realiza tenemos:

CODEC.- Realiza el análisis del Codec inyectando una onda seno de 1 KHz. en el canal derecho (amarillo) y una onda seno de 2 k.o. en el canal izquierdo (azul).

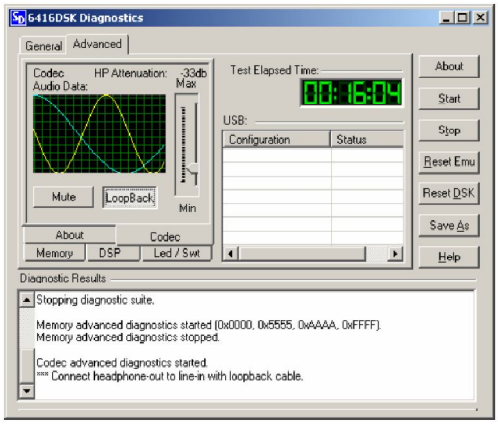

Figura C.3 Diagnostico Avanzado CODEC.

MEMORIA. - Analiza la memoria Interna y externa.

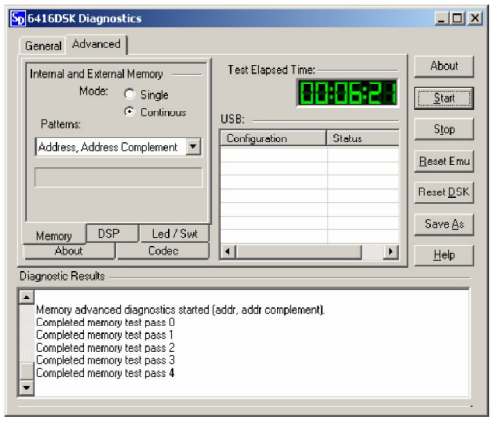

Figura C.4 Diagnostico Avanzado Memoria.

DSP.- Este diagnostico ejecuta cuatro análisis en el periférico del chip. El análisis temporizado muestra un LED parpadeante moviéndose de izquierda a derecha y regreso. El movimiento del LED permite al usuario verificar que el timer está trabajando. La DMA usa el DMA del chip para copiar datos de un buffer a otro. El diagnostico McBSP pasa un conjunto de datos a través

del puerto Serial 0 y 1 (McBSP es una puerto serial Texas Instruments) usando el mecanismo de internal loop-back y generador de frecuencia.

Características de análisis es indicar si un problema con el DSP ocurre.

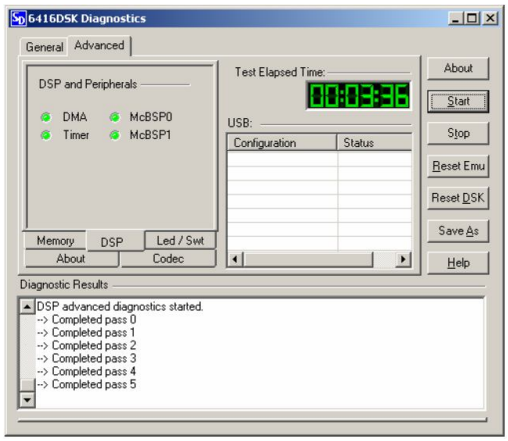

Figura C.5 Diagnostico Avanzado DSP.

LED/SWITCH.- Realiza el diagnostico del estado del DIP switch y constantemente actualiza el chequeo de los LED de la tarjeta. Los LED tienen la misma polaridad de los switches.

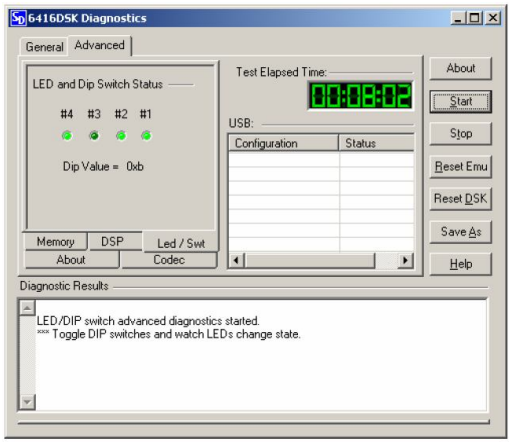

Figura C.6 Diagnostico Avanzado Led/Swt.

ANEXO D

Se presenta los valores de los centroides formados con un vector de 50 datos, diez repeticiones de cada cinco palabra.

Dependiendo de la formula $M = 2^a$, determina el tamaño optimo del codebook.

a=2;

-23.0000	-1.0000	13.2727	-17.3636	1.8182	21.5455	-32.9091	14.3636	31.6364	-75.1818
-22.2000	-1.0000	12.8667	-17.4000	-0.1333	26.7333	-34.2000	0.1333	60.4000	-83.5333
-22.4000	-1.0000	13.4000	-22.3333	7.6000	28.0000	-58.9333	39.5333	55.6667	-177.1333
-23.4444	-1.0000	14.3333	-28.2222	17.6667	27.0000	-88.3333	104.4444	9.8889	-275.7778

a=3;

-25.2000	-1.0000	14.0000	-15.2000	-3.6000	22.4000	-19.0000	-8.4000	35.4000	-28.6000
-25.4000	-1.0000	14.4000	-17.6000	-2.6000	27.6000	-26.6000	-10.2000	54.6000	-47.4000
-19.3333	-1.0000	11.5000	-16.5000	2.5000	22.1667	-36.1667	13.3333	45.0000	-92.6667
-21.1667	-1.0000	12.3333	-17.6667	0.6667	27.6667	-38.6667	2.5000	69.3333	-104.0000
-22.8000	-1.0000	13.6000	-21.2000	7.6000	22.8000	-50.6000	40.0000	31.0000	-133.6000
-21.8333	-1.0000	13.0833	-21.6667	6.5833	28.9167	-57.6667	33.5000	64.6667	-178.5833
-23.7500	-1.0000	14.2500	-24.7500	10.7500	27.7500	-67.2500	58.0000	42.5000	-199.7500
-23.7143	-1.0000	14.5714	-29.5714	20.4286	25.7143	-94.7143	122.4286	-8.4286	-292.7143

a=4;

-27.0000	-1.0000	15.0000	-14.5000	-6.5000	23.0000	-13.0000	-16.5000	33.5000	-10.5000
-25.0000	-1.0000	14.0000	-15.5000	-5.0000	26.5000	-18.5000	-19.0000	49.0000	-20.5000
-24.0000	-1.0000	13.5000	-16.5000	-1.5000	24.0000	-25.7500	-4.2500	45.0000	-48.5000
-22.5000	-1.0000	13.0000	-17.0000	-0.7500	26.0000	-31.0000	-2.7500	57.0000	-70.0000
-21.0000	-1.0000	12.5000	-17.7500	2.7500	23.5000	-38.2500	15.0000	45.5000	-96.5000
-22.5000	-1.0000	13.0000	-20.0000	7.0000	20.0000	-45.0000	36.5000	22.5000	-109.5000
-20.6000	-1.0000	12.0000	-17.4000	0.8000	27.2000	-38.8000	3.0000	69.4000	-106.0000
-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000
-22.2000	-1.0000	13.4000	-22.0000	8.8000	24.0000	-55.6000	47.2000	31.8000	-153.4000
-22.4286	-1.0000	13.4286	-21.8571	6.2857	29.4286	-57.2857	31.5714	65.7143	-174.2857
-21.7500	-1.0000	13.0000	-22.2500	8.2500	27.2500	-60.7500	44.2500	54.2500	-189.5000
-22.6667	-1.0000	13.6667	-23.3333	7.6667	31.6667	-64.6667	39.0000	74.0000	-209.0000
-22.0000	-1.0000	13.5000	-27.5000	19.5000	22.5000	-87.0000	117.0000	-19.0000	-258.0000
-23.3333	-1.0000	14.3333	-29.3333	21.0000	23.6667	-93.3333	126.6667	-22.0000	-278.0000
-27.9000	-27.9000	-27.9000	-27.9000	-27.9000	-27.9000	-27.9000	-27.9000	-27.9000	-27.9000
-32.2000	-32.2000	-32.2000	-32.2000	-32.2000	-32.2000	-32.2000	-32.2000	-32.2000	-32.2000

a=5;

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-23.2000	-1.0000	13.2000	-16.7000	-1.2000	25.1000	-28.3000	-3.8000	51.1000	-58.8000
-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000
-20.0000	-1.0000	11.6667	-17.3333	2.6667	23.3333	-38.6667	14.0000	49.3333	-102.3333
-20.5000	-1.0000	12.0000	-17.2500	1.0000	27.0000	-39.0000	3.7500	69.2500	-107.5000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-26.3333	-1.0000	14.6667	-14.6667	-6.0000	23.6667	-14.3333	-16.6667	36.6667	-13.0000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-21.8000	-1.0000	13.2000	-20.2000	7.2000	21.6000	-47.8000	38.2000	27.8000	-124.4000
-22.7917	-1.0000	13.7500	-24.5417	11.3750	27.6250	-69.9583	63.8750	38.5000	-214.1250
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

a=6;

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-26.4000	-1.0000	14.8000	-15.8000	-5.2000	25.4000	-18.0000	-15.8000	43.0000	-22.0000
-2.1000	-2.1000	-2.1000	-2.1000	-2.1000	-2.1000	-2.1000	-2.1000	-2.1000	-2.1000
-24.3333	-1.0000	13.6667	-17.0000	-0.3333	22.6667	-27.6667	2.3333	39.0000	-55.6667
-4.6000	-4.6000	-4.6000	-4.6000	-4.6000	-4.6000	-4.6000	-4.6000	-4.6000	-4.6000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-19.0000	-1.0000	11.0000	-15.5000	1.0000	22.5000	-33.0000	5.5000	51.5000	-84.5000
-4.7000	-4.7000	-4.7000	-4.7000	-4.7000	-4.7000	-4.7000	-4.7000	-4.7000	-4.7000
-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000	-6.3000
-6.8000	-6.8000	-6.8000	-6.8000	-6.8000	-6.8000	-6.8000	-6.8000	-6.8000	-6.8000
-19.0000	-1.0000	11.0000	-16.5000	2.0000	24.5000	-38.5000	9.0000	61.0000	-109.5000
-22.0000	-1.0000	13.0000	-18.0000	0	29.5000	-39.5000	-1.5000	77.5000	-105.5000
-19.5000	-1.0000	12.0000	-17.5000	4.5000	20.5000	-39.5000	24.5000	32.5000	-100.5000
-22.5000	-1.0000	13.0000	-20.0000	7.0000	20.0000	-45.0000	36.5000	22.5000	-109.5000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000	-10.8000
-22.0000	-1.0000	13.3333	-21.0000	7.3333	24.0000	-51.6667	39.0000	37.3333	-143.3333
0	0	0	0	0	0	0	0	0	0
-16.6000	-16.6000	-16.6000	-16.6000	-16.6000	-16.6000	-16.6000	-16.6000	-16.6000	-16.6000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-22.6000	-1.0000	13.6000	-21.6000	5.0000	30.6000	-55.2000	24.6000	72.8000	-167.8000
-22.0000	-1.0000	13.0000	-21.5000	5.5000	30.5000	-57.0000	26.5000	76.5000	-179.5000
-14.8000	-14.8000	-14.8000	-14.8000	-14.8000	-14.8000	-14.8000	-14.8000	-14.8000	-14.8000
-23.5000	-1.0000	14.0000	-24.0000	11.0000	25.5000	-65.0000	59.5000	33.0000	-190.5000
-20.0000	-1.0000	12.0000	-21.0000	8.0000	26.0000	-59.0000	43.0000	55.0000	-190.5000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-23.0000	-1.0000	14.0000	-23.5000	8.0000	31.5000	-64.5000	40.5000	69.5000	-204.5000
-15.8000	-15.8000	-15.8000	-15.8000	-15.8000	-15.8000	-15.8000	-15.8000	-15.8000	-15.8000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-22.3000	-22.3000	-22.3000	-22.3000	-22.3000	-22.3000	-22.3000	-22.3000	-22.3000	-22.3000
-23.8333	-1.0000	14.6667	-30.0000	21.1667	25.5000	-96.8333	127.6667	-13.1667	-298.5000
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

ANEXO F.

Código de la programación principal para el reconocimiento de voz, con el uso del toolbox GUI.

```
function varargout = recvoc(varargin)
% RECVOC M-file for recvoc.fig
%      RECVOC, by itself, creates a new RECVOC or raises the existing
%      singleton*.
%
%      H = RECVOC returns the handle to a new RECVOC or the handle to
%      the existing singleton*.
%
%      RECVOC('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RECVOC.M with the given input arguments.
%
%      RECVOC('Property','Value',...) creates a new RECVOC or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before recvoc_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to recvoc_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help recvoc
% Last Modified by GUIDE v2.5 20-May-2011 12:38:48
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @recvoc_OpeningFcn, ...
                   'gui_OutputFcn',  @recvoc_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --------------------------------------------------------------------
%PROGRAMA PRINCIPAL
% --- Executes just before recvoc is made visible.
```

```
function recvoc_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to recvoc (see VARARGIN)

% Choose default command line output for recvoc
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=1;
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=2;
% Hint: get(hObject,'Value') returns toggle state of radiobutton2
switch a
   case 1;
      load ayuda.mat;
      load bao.mat;
      load camino.mat;
      load casa.mat;
      load cobija.mat;
      load comer.mat;
      load dame.mat;
      load enfermo.mat;
      load frio.mat;
      load gracias.mat;

      handles.feature_f0=f0;
      handles.feature_f1=f1;
      handles.feature_f2=f2;
      handles.feature_f3=f3;
      handles.feature_f4=f4;
      handles.feature_f5=f5;
      handles.feature_f6=f6;
      handles.feature_f7=f7;
      handles.feature_f8=f8;
      handles.feature_f9=f9;

      handles.feature_y0=y0;
      handles.feature_y1=y1;
      handles.feature_y2=y2;
      handles.feature_y3=y3;
      handles.feature_y4=y4;
```

```
        handles.feature_y5=y5;
        handles.feature_y6=y6;
        handles.feature_y7=y7;
        handles.feature_y8=y8;
        handles.feature_y9=y9;
    case 2;
        load hambre.mat;
        load hijo.mat;
        load maria.mat;
        load quiero.mat;
        load ropa.mat;
        load sed.mat;
        load tengo.mat;
        load ven.mat;
        load yo.mat;
        load zapato.mat;

        handles.feature_f0=f10;
        handles.feature_f1=f11;
        handles.feature_f2=f12;
        handles.feature_f3=f13;
        handles.feature_f4=f14;
        handles.feature_f5=f15;
        handles.feature_f6=f16;
        handles.feature_f7=f17;
        handles.feature_f8=f18;
        handles.feature_f9=f19;

        handles.feature_y0=y10;
        handles.feature_y1=y11;
        handles.feature_y2=y12;
        handles.feature_y3=y13;
        handles.feature_y4=y14;
        handles.feature_y5=y15;
        handles.feature_y6=y16;
        handles.feature_y7=y17;
        handles.feature_y8=y18;
        handles.feature_y9=y19;
end
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes recvoc wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = recvoc_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
```

```
varargout{1} = handles.output;


% --- Executes on button press in Grabar.
function Grabar_Callback(hObject, eventdata, handles)
% hObject    handle to Grabar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% obtener la palabra del microfono
Fs = 16000; % Sampling Frequency (Hz)
n = 20;
Nseconds = 2; % largo de la señal de voz
y = wavrecord(Nseconds*Fs, Fs, 1)';
rec1=reconociendovoz(y);
handles.voz_grabada = y;
handles.feature_datos=rec1;
plot(y);
% msgbox('Grabacion terminada');
guidata(hObject, handles);
%load rec.mat;
%handles.feature_voz=f;

% --- Executes on button press in Identificar.
function Identificar_Callback(hObject, eventdata, handles)
% hObject    handle to Identificar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
rec1=handles.feature_datos;
y0=handles.feature_y0;
y1=handles.feature_y1;
y2=handles.feature_y2;
y3=handles.feature_y3;
y4=handles.feature_y4;
y5=handles.feature_y5;
y6=handles.feature_y6;
y7=handles.feature_y7;
y8=handles.feature_y8;
y9=handles.feature_y9;

f0=handles.feature_f0;
f1=handles.feature_f1;
f2=handles.feature_f2;
f3=handles.feature_f3;
f4=handles.feature_f4;
f5=handles.feature_f5;
f6=handles.feature_f6;
f7=handles.feature_f7;
f8=handles.feature_f8;
f9=handles.feature_f9;
op=a;
switch op
```

```
   case 1
      d1=distancia(f0,rec1);
      d2=distancia(f1,rec1);
      d3=distancia(f2,rec1);
      d4=distancia(f3,rec1);
      d5=distancia(f4,rec1);
      d6=distancia(f5,rec1);
      d7=distancia(f6,rec1);
      d8=distancia(f7,rec1);
      d9=distancia(f8,rec1);
      d10=distancia(f9,rec1);

      Ds1=size(d1);Ds2=size(d2);
      Ds3=size(d3);Ds4=size(d4);
      Ds5=size(d5);Ds6=size(d6);
      Ds7=size(d7);Ds8=size(d8);
      Ds9=size(d9);Ds10=size(d10);


dista1=cat(2,Ds1(1),Ds2(1),Ds3(1),Ds4(1),Ds5(1),Ds6(1),Ds7(1),Ds8(1),Ds9(1),Ds10(1));

dista2=cat(2,Ds1(2),Ds2(2),Ds3(2),Ds4(2),Ds5(2),Ds6(2),Ds7(2),Ds8(2),Ds9(2),Ds10(2));

      minds1=min(dista1);
      minds2=min(dista2);

      d1=d1(1:minds1,1:minds2);
      d2=d2(1:minds1,1:minds2);
      d3=d3(1:minds1,1:minds2);
      d4=d4(1:minds1,1:minds2);
      d5=d5(1:minds1,1:minds2);
      d6=d6(1:minds1,1:minds2);
      d7=d7(1:minds1,1:minds2);
      d8=d8(1:minds1,1:minds2);
      d9=d9(1:minds1,1:minds2);
      d10=d10(1:minds1,1:minds2);

      %MAXIMOS.....
      mzds1=.... max(d1);
      mzds2=.... max(d2);
      mzds3=.... max(d3);
      mzds4=.... max(d4);
      mzds5=.... max(d5);
      mzds6=.... max(d6);
      mzds7=.... max(d7);
      mzds8=.... max(d8);
      mzds9=.... max(d9);
      mzds10=.... max(d10);

      %MIMINOS.....
      mzdn1=.... min(d1);
```

```
mzdn2=…. min(d2);
mzdn3=…. min(d3);
mzdn4=…. min(d4);
mzdn5=…. min(d5);
mzdn6=…. min(d6);
mzdn7=…. min(d7);
mzdn8=…. min(d8);
mzdn9=…. min(d9);
mzdn10=…. min(d10);
%MODA.....
mzdm1=…. mode(d1);
mzdm2=…. mode(d2);
mzdm3=…. mode(d3);
mzdm4=…. mode(d4);
mzdm5=…. mode(d5);
mzdm6=…. mode(d6);
mzdm7=…. mode(d7);
mzdm8=…. mode(d8);
mzdm9=…. mode(d9);
mzdm10=…. mode(d10);

%1111111111111111111111111111111111111111111111111111
%Maximos=======================================
undis11=(mzds1<mzds2);
undis12=(mzds1<mzds3);
undis13=(mzds1<mzds4);
undis14=(mzds1<mzds5);
undis15=(mzds1<mzds6);
undis16=(mzds1<mzds7);
undis17=(mzds1<mzds8);
undis18=(mzds1<mzds9);
undis19=(mzds1<mzds10);
total11=length(find(undis11==1));
total12=length(find(undis12==1));
total13=length(find(undis13==1));
total14=length(find(undis14==1));
total15=length(find(undis15==1));
total16=length(find(undis16==1));
total17=length(find(undis17==1));
total18=length(find(undis18==1));
total19=length(find(undis19==1));
t1=total11+total12+total13+total14+total15+total16+total17+total18+total19;
%Minimos=======================================
undin11=(mzdn1<mzdn2);
undin12=(mzdn1<mzdn3);
undin13=(mzdn1<mzdn4);
undin14=(mzdn1<mzdn5);
undin15=(mzdn1<mzdn6);
undin16=(mzdn1<mzdn7);
undin17=(mzdn1<mzdn8);
undin18=(mzdn1<mzdn9);
```

```
    undin19=(mzdn1<mzdn10);
    totan11=length(find(undin11==1));
    totan12=length(find(undin12==1));
    totan13=length(find(undin13==1));
    totan14=length(find(undin14==1));
    totan15=length(find(undin15==1));
    totan16=length(find(undin16==1));
    totan17=length(find(undin17==1));
    totan18=length(find(undin18==1));
    totan19=length(find(undin19==1));
    tn1=totan11+totan12+totan13+totan14+totan15+totan16+totan17+totan18+totan19;
    %Moda=======================================
    undim11=(mzdm1<mzdm2);
    undim12=(mzdm1<mzdm3);
    undim13=(mzdm1<mzdm4);
    undim14=(mzdm1<mzdm5);
    undim15=(mzdm1<mzdm6);
    undim16=(mzdm1<mzdm7);
    undim17=(mzdm1<mzdm8);
    undim18=(mzdm1<mzdm9);
    undim19=(mzdm1<mzdm10);
    totam11=length(find(undim11==1));
    totam12=length(find(undim12==1));
    totam13=length(find(undim13==1));
    totam14=length(find(undim14==1));
    totam15=length(find(undim15==1));
    totam16=length(find(undim16==1));
    totam17=length(find(undim17==1));
    totam18=length(find(undim18==1));
    totam19=length(find(undim19==1));

tm1=totam11+totam12+totam13+totam14+totam15+totam16+totam17+totam18+totam19;

% 22222222222222222222222222222222222222222222222222
    %Maximos=====================================
    undis21=(mzds2<mzds1);
    undis22=(mzds2<mzds3);
    undis23=(mzds2<mzds4);
    undis24=(mzds2<mzds5);
    undis25=(mzds2<mzds6);
    undis26=(mzds2<mzds7);
    undis27=(mzds2<mzds8);
    undis28=(mzds2<mzds9);
    undis29=(mzds2<mzds10);
    total21=length(find(undis21==1));
    total22=length(find(undis22==1));
    total23=length(find(undis23==1));
    total24=length(find(undis24==1));
    total25=length(find(undis25==1));
    total26=length(find(undis26==1));
    total27=length(find(undis27==1));
```

```
total28=length(find(undis28==1));
total29=length(find(undis29==1));
t2=total21+total22+total23+total24+total25+total26+total27+total28+total29;
%Minimos=======================================
undin21=(mzdn2<mzdn1);
undin22=(mzdn2<mzdn3);
undin23=(mzdn2<mzdn4);
undin24=(mzdn2<mzdn5);
undin25=(mzdn2<mzdn6);
undin26=(mzdn2<mzdn7);
undin27=(mzdn2<mzdn8);
undin28=(mzdn2<mzdn9);
undin29=(mzdn2<mzdn10);
totan21=length(find(undin21==1));
totan22=length(find(undin22==1));
totan23=length(find(undin23==1));
totan24=length(find(undin24==1));
totan25=length(find(undin25==1));
totan26=length(find(undin26==1));
totan27=length(find(undin27==1));
totan28=length(find(undin28==1));
totan29=length(find(undin29==1));
tn2=totan21+totan22+totan23+totan24+totan25+totan26+totan27+totan28+totan29;
%Moda=======================================
undim21=(mzdm2<mzdm1);
undim22=(mzdm2<mzdm3);
undim23=(mzdm2<mzdm4);
undim24=(mzdm2<mzdm5);
undim25=(mzdm2<mzdm6);
undim26=(mzdm2<mzdm7);
undim27=(mzdm2<mzdm8);
undim28=(mzdm2<mzdm9);
undim29=(mzdm2<mzdm10);
totam21=length(find(undim21==1));
totam22=length(find(undim22==1));
totam23=length(find(undim23==1));
totam24=length(find(undim24==1));
totam25=length(find(undim25==1));
totam26=length(find(undim26==1));
totam27=length(find(undim27==1));
totam28=length(find(undim28==1));
totam29=length(find(undim29==1));

tm2=totam21+totam22+totam23+totam24+totam25+totam26+totam27+totam28+totam29;

%3333333333333333333333333333333333333333333333333333333
%Maximos=======================================
undis31=(mzds3<mzds1);
undis32=(mzds3<mzds2);
undis33=(mzds3<mzds4);
undis34=(mzds3<mzds5);
```

```
undis35=(mzds3<mzds6);
undis36=(mzds3<mzds7);
undis37=(mzds3<mzds8);
undis38=(mzds3<mzds9);
undis39=(mzds3<mzds10);
total31=length(find(undis31==1));
total32=length(find(undis32==1));
total33=length(find(undis33==1));
total34=length(find(undis34==1));
total35=length(find(undis35==1));
total36=length(find(undis36==1));
total37=length(find(undis37==1));
total38=length(find(undis38==1));
total39=length(find(undis39==1));
t3=total31+total32+total33+total34+total35+total36+total37+total38+total39;
%Minimos=======================================
undin31=(mzdn3<mzdn1);
undin32=(mzdn3<mzdn2);
undin33=(mzdn3<mzdn4);
undin34=(mzdn3<mzdn5);
undin35=(mzdn3<mzdn6);
undin36=(mzdn3<mzdn7);
undin37=(mzdn3<mzdn8);
undin38=(mzdn3<mzdn9);
undin39=(mzdn3<mzdn10);
totan31=length(find(undin31==1));
totan32=length(find(undin32==1));
totan33=length(find(undin33==1));
totan34=length(find(undin34==1));
totan35=length(find(undin35==1));
totan36=length(find(undin36==1));
totan37=length(find(undin37==1));
totan38=length(find(undin38==1));
totan39=length(find(undin39==1));
tn3=totan31+totan32+totan33+totan34+totan35+totan36+totan37+totan38+totan39;
%Moda==========================================
undim31=(mzdm3<mzdm1);
undim32=(mzdm3<mzdm2);
undim33=(mzdm3<mzdm4);
undim34=(mzdm3<mzdm5);
undim35=(mzdm3<mzdm6);
undim36=(mzdm3<mzdm7);
undim37=(mzdm3<mzdm8);
undim38=(mzdm3<mzdm9);
undim39=(mzdm3<mzdm10);
totam31=length(find(undim31==1));
totam32=length(find(undim32==1));
totam33=length(find(undim33==1));
totam34=length(find(undim34==1));
totam35=length(find(undim35==1));
totam36=length(find(undim36==1));
```

```
    totam37=length(find(undim37==1));
    totam38=length(find(undim38==1));
    totam39=length(find(undim39==1));

tm3=totam31+totam32+totam33+totam34+totam35+totam36+totam37+totam38+totam39;

%4444444444444444444444444444444444444444444444444
    %Maximos=======================================
    undis41=(mzds4<mzds1);
    undis42=(mzds4<mzds2);
    undis43=(mzds4<mzds3);
    undis44=(mzds4<mzds5);
    undis45=(mzds4<mzds6);
    undis46=(mzds4<mzds7);
    undis47=(mzds4<mzds8);
    undis48=(mzds4<mzds9);
    undis49=(mzds4<mzds10);
    total41=length(find(undis41==1));
    total42=length(find(undis42==1));
    total43=length(find(undis43==1));
    total44=length(find(undis44==1));
    total45=length(find(undis45==1));
    total46=length(find(undis46==1));
    total47=length(find(undis47==1));
    total48=length(find(undis48==1));
    total49=length(find(undis49==1));
    t4=total41+total42+total43+total44+total45+total46+total47+total48+total49;
    %Minimos=======================================
    undin41=(mzdn4<mzdn1);
    undin42=(mzdn4<mzdn2);
    undin43=(mzdn4<mzdn3);
    undin44=(mzdn4<mzdn5);
    undin45=(mzdn4<mzdn6);
    undin46=(mzdn4<mzdn7);
    undin47=(mzdn4<mzdn8);
    undin48=(mzdn4<mzdn9);
    undin49=(mzdn4<mzdn10);
    totan41=length(find(undin41==1));
    totan42=length(find(undin42==1));
    totan43=length(find(undin43==1));
    totan44=length(find(undin44==1));
    totan45=length(find(undin45==1));
    totan46=length(find(undin46==1));
    totan47=length(find(undin47==1));
    totan48=length(find(undin48==1));
    totan49=length(find(undin49==1));
    tn4=totan41+totan42+totan43+totan44+totan45+totan46+totan47+totan48+totan49;
    %Moda==========================================
    undim41=(mzdm4<mzdm1);
    undim42=(mzdm4<mzdm2);
    undim43=(mzdm4<mzdm3);
```

```
        undim44=(mzdm4<mzdm5);
        undim45=(mzdm4<mzdm6);
        undim46=(mzdm4<mzdm7);
        undim47=(mzdm4<mzdm8);
        undim48=(mzdm4<mzdm9);
        undim49=(mzdm4<mzdm10);
        totam41=length(find(undim41==1));
        totam42=length(find(undim42==1));
        totam43=length(find(undim43==1));
        totam44=length(find(undim44==1));
        totam45=length(find(undim45==1));
        totam46=length(find(undim46==1));
        totam47=length(find(undim47==1));
        totam48=length(find(undim48==1));
        totam49=length(find(undim49==1));

tm4=totam41+totam42+totam43+totam44+totam45+totam46+totam47+totam48+totam49;

%555555555555555555555555555555555555555555555555555555555
        %Maximos=======================================
        undis51=(mzds5<mzds1);
        undis52=(mzds5<mzds2);
        undis53=(mzds5<mzds3);
        undis54=(mzds5<mzds4);
        undis55=(mzds5<mzds6);
        undis56=(mzds5<mzds7);
        undis57=(mzds5<mzds8);
        undis58=(mzds5<mzds9);
        undis59=(mzds5<mzds10);
        total51=length(find(undis51==1));
        total52=length(find(undis52==1));
        total53=length(find(undis53==1));
        total54=length(find(undis54==1));
        total55=length(find(undis55==1));
        total56=length(find(undis56==1));
        total57=length(find(undis57==1));
        total58=length(find(undis58==1));
        total59=length(find(undis59==1));
        t5=total51+total52+total53+total54+total55+total56+total57+total58+total59;
        %Minimos=======================================
        undin51=(mzdn5<mzdn1);
        undin52=(mzdn5<mzdn2);
        undin53=(mzdn5<mzdn3);
        undin54=(mzdn5<mzdn4);
        undin55=(mzdn5<mzdn6);
        undin56=(mzdn5<mzdn7);
        undin57=(mzdn5<mzdn8);
        undin58=(mzdn5<mzdn9);
        undin59=(mzdn5<mzdn10);
        totan51=length(find(undin51==1));
        totan52=length(find(undin52==1));
```

```
totan53=length(find(undin53==1));
totan54=length(find(undin54==1));
totan55=length(find(undin55==1));
totan56=length(find(undin56==1));
totan57=length(find(undin57==1));
totan58=length(find(undin58==1));
totan59=length(find(undin59==1));
tn5=totan51+totan52+totan53+totan54+totan55+totan56+totan57+totan58+totan59;
%Moda=======================================
undim51=(mzdm5<mzdm1);
undim52=(mzdm5<mzdm2);
undim53=(mzdm5<mzdm3);
undim54=(mzdm5<mzdm4);
undim55=(mzdm5<mzdm6);
undim56=(mzdm5<mzdm7);
undim57=(mzdm5<mzdm8);
undim58=(mzdm5<mzdm9);
undim59=(mzdm5<mzdm10);
totam51=length(find(undim51==1));
totam52=length(find(undim52==1));
totam53=length(find(undim53==1));
totam54=length(find(undim54==1));
totam55=length(find(undim55==1));
totam56=length(find(undim56==1));
totam57=length(find(undim57==1));
totam58=length(find(undim58==1));
totam59=length(find(undim59==1));

tm5=totam51+totam52+totam53+totam54+totam55+totam56+totam57+totam58+totam59;

%6666666666666666666666666666666666666666666666666666666666666666
%Maximos=======================================
undis61=(mzds6<mzds1);
undis62=(mzds6<mzds2);
undis63=(mzds6<mzds3);
undis64=(mzds6<mzds4);
undis65=(mzds6<mzds5);
undis66=(mzds6<mzds7);
undis67=(mzds6<mzds8);
undis68=(mzds6<mzds9);
undis69=(mzds6<mzds10);
total61=length(find(undis61==1));
total62=length(find(undis62==1));
total63=length(find(undis63==1));
total64=length(find(undis64==1));
total65=length(find(undis65==1));
total66=length(find(undis66==1));
total67=length(find(undis67==1));
total68=length(find(undis68==1));
total69=length(find(undis69==1));
t6=total61+total62+total63+total64+total65+total66+total67+total68+total69;
```

```
%Minimos=======================================
undin61=(mzdn6<mzdn1);
undin62=(mzdn6<mzdn2);
undin63=(mzdn6<mzdn3);
undin64=(mzdn6<mzdn4);
undin65=(mzdn6<mzdn5);
undin66=(mzdn6<mzdn7);
undin67=(mzdn6<mzdn8);
undin68=(mzdn6<mzdn9);
undin69=(mzdn6<mzdn10);
totan61=length(find(undin61==1));
totan62=length(find(undin62==1));
totan63=length(find(undin63==1));
totan64=length(find(undin64==1));
totan65=length(find(undin65==1));
totan66=length(find(undin66==1));
totan67=length(find(undin67==1));
totan68=length(find(undin68==1));
totan69=length(find(undin69==1));
tn6=totan61+totan62+totan63+totan64+totan65+totan66+totan67+totan68+totan69;
%Moda=========================================
undim61=(mzdm6<mzdm1);
undim62=(mzdm6<mzdm2);
undim63=(mzdm6<mzdm3);
undim64=(mzdm6<mzdm4);
undim65=(mzdm6<mzdm5);
undim66=(mzdm6<mzdm7);
undim67=(mzdm6<mzdm8);
undim68=(mzdm6<mzdm9);
undim69=(mzdm6<mzdm10);
totam61=length(find(undim61==1));
totam62=length(find(undim62==1));
totam63=length(find(undim63==1));
totam64=length(find(undim64==1));
totam65=length(find(undim65==1));
totam66=length(find(undim66==1));
totam67=length(find(undim67==1));
totam68=length(find(undim68==1));
totam69=length(find(undim69==1));

tm6=totam61+totam62+totam63+totam64+totam65+totam66+totam67+totam68+totam69;

%7777777777777777777777777777777777777777777777777777777777777777777777
%Maximos=======================================
undis71=(mzds7<mzds1);
undis72=(mzds7<mzds2);
undis73=(mzds7<mzds3);
undis74=(mzds7<mzds4);
undis75=(mzds7<mzds5);
undis76=(mzds7<mzds6);
undis77=(mzds7<mzds8);
```

```
undis78=(mzds7<mzds9);
undis79=(mzds7<mzds10);
total71=length(find(undis71==1));
total72=length(find(undis72==1));
total73=length(find(undis73==1));
total74=length(find(undis74==1));
total75=length(find(undis75==1));
total76=length(find(undis76==1));
total77=length(find(undis77==1));
total78=length(find(undis78==1));
total79=length(find(undis79==1));
t7=total71+total72+total73+total74+total75+total76+total77+total78+total79;
%Minimos=======================================
undin71=(mzdn7<mzdn1);
undin72=(mzdn7<mzdn2);
undin73=(mzdn7<mzdn3);
undin74=(mzdn7<mzdn4);
undin75=(mzdn7<mzdn5);
undin76=(mzdn7<mzdn6);
undin77=(mzdn7<mzdn8);
undin78=(mzdn7<mzdn9);
undin79=(mzdn7<mzdn10);
totan71=length(find(undin71==1));
totan72=length(find(undin72==1));
totan73=length(find(undin73==1));
totan74=length(find(undin74==1));
totan75=length(find(undin75==1));
totan76=length(find(undin76==1));
totan77=length(find(undin77==1));
totan78=length(find(undin78==1));
totan79=length(find(undin79==1));
tn7=totan71+totan72+totan73+totan74+totan75+totan76+totan77+totan78+totan79;
%Moda=========================================
undim71=(mzdm7<mzdm1);
undim72=(mzdm7<mzdm2);
undim73=(mzdm7<mzdm3);
undim74=(mzdm7<mzdm4);
undim75=(mzdm7<mzdm5);
undim76=(mzdm7<mzdm6);
undim77=(mzdm7<mzdm8);
undim78=(mzdm7<mzdm9);
undim79=(mzdm7<mzdm10);
totam71=length(find(undim71==1));
totam72=length(find(undim72==1));
totam73=length(find(undim73==1));
totam74=length(find(undim74==1));
totam75=length(find(undim75==1));
totam76=length(find(undim76==1));
totam77=length(find(undim77==1));
totam78=length(find(undim78==1));
totam79=length(find(undim79==1));
```

```
tm7=totam71+totam72+totam73+totam74+totam75+totam76+totam77+totam78+totam79;

%888888888888888888888888888888888888888888888888888888888888
    %Maximos=======================================
    undis81=(mzds8<mzds1);
    undis82=(mzds8<mzds2);
    undis83=(mzds8<mzds3);
    undis84=(mzds8<mzds4);
    undis85=(mzds8<mzds5);
    undis86=(mzds8<mzds6);
    undis87=(mzds8<mzds7);
    undis88=(mzds8<mzds9);
    undis89=(mzds8<mzds10);
    total81=length(find(undis81==1));
    total82=length(find(undis82==1));
    total83=length(find(undis83==1));
    total84=length(find(undis84==1));
    total85=length(find(undis85==1));
    total86=length(find(undis86==1));
    total87=length(find(undis87==1));
    total88=length(find(undis88==1));
    total89=length(find(undis89==1));
    t8=total81+total82+total83+total84+total85+total86+total87+total88+total89;
    %Minimos=======================================
    undin81=(mzdn8<mzdn1);
    undin82=(mzdn8<mzdn2);
    undin83=(mzdn8<mzdn3);
    undin84=(mzdn8<mzdn4);
    undin85=(mzdn8<mzdn5);
    undin86=(mzdn8<mzdn6);
    undin87=(mzdn8<mzdn7);
    undin88=(mzdn8<mzdn9);
    undin89=(mzdn8<mzdn10);
    totan81=length(find(undin81==1));
    totan82=length(find(undin82==1));
    totan83=length(find(undin83==1));
    totan84=length(find(undin84==1));
    totan85=length(find(undin85==1));
    totan86=length(find(undin86==1));
    totan87=length(find(undin87==1));
    totan88=length(find(undin88==1));
    totan89=length(find(undin89==1));
    tn8=totan81+totan82+totan83+totan84+totan85+totan86+totan87+totan88+totan89;
    %Moda=======================================
    undim81=(mzdm8<mzdm1);
    undim82=(mzdm8<mzdm2);
    undim83=(mzdm8<mzdm3);
    undim84=(mzdm8<mzdm4);
    undim85=(mzdm8<mzdm5);
    undim86=(mzdm8<mzdm6);
```

```
    undim87=(mzdm8<mzdm7);
    undim88=(mzdm8<mzdm9);
    undim89=(mzdm8<mzdm10);
    totam81=length(find(undim81==1));
    totam82=length(find(undim82==1));
    totam83=length(find(undim83==1));
    totam84=length(find(undim84==1));
    totam85=length(find(undim85==1));
    totam86=length(find(undim86==1));
    totam87=length(find(undim87==1));
    totam88=length(find(undim88==1));
    totam89=length(find(undim89==1));

tm8=totam81+totam82+totam83+totam84+totam85+totam86+totam87+totam88+totam89;

%99999999999999999999999999999999999999999999999999999999
    %Maximos=======================================
    undis91=(mzds9<mzds1);
    undis92=(mzds9<mzds2);
    undis93=(mzds9<mzds3);
    undis94=(mzds9<mzds4);
    undis95=(mzds9<mzds5);
    undis96=(mzds9<mzds6);
    undis97=(mzds9<mzds7);
    undis98=(mzds9<mzds8);
    undis99=(mzds9<mzds10);
    total91=length(find(undis91==1));
    total92=length(find(undis92==1));
    total93=length(find(undis93==1));
    total94=length(find(undis94==1));
    total95=length(find(undis95==1));
    total96=length(find(undis96==1));
    total97=length(find(undis97==1));
    total98=length(find(undis98==1));
    total99=length(find(undis99==1));
    t9=total91+total92+total93+total94+total95+total96+total97+total98+total99;
    %Minimos=======================================
    undin91=(mzdn9<mzdn1);
    undin92=(mzdn9<mzdn2);
    undin93=(mzdn9<mzdn3);
    undin94=(mzdn9<mzdn4);
    undin95=(mzdn9<mzdn5);
    undin96=(mzdn9<mzdn6);
    undin97=(mzdn9<mzdn7);
    undin98=(mzdn9<mzdn8);
    undin99=(mzdn9<mzdn10);
    totan91=length(find(undin91==1));
    totan92=length(find(undin92==1));
    totan93=length(find(undin93==1));
    totan94=length(find(undin94==1));
    totan95=length(find(undin95==1));
```

```
totan96=length(find(undin96==1));
totan97=length(find(undin97==1));
totan98=length(find(undin98==1));
totan99=length(find(undin99==1));
tn9=totan91+totan92+totan93+totan94+totan95+totan96+totan97+totan98+totan99;
%Moda=========================================
undim91=(mzdm9<mzdm1);
undim92=(mzdm9<mzdm2);
undim93=(mzdm9<mzdm3);
undim94=(mzdm9<mzdm4);
undim95=(mzdm9<mzdm5);
undim96=(mzdm9<mzdm6);
undim97=(mzdm9<mzdm7);
undim98=(mzdm9<mzdm8);
undim99=(mzdm9<mzdm10);
totam91=length(find(undim91==1));
totam92=length(find(undim92==1));
totam93=length(find(undim93==1));
totam94=length(find(undim94==1));
totam95=length(find(undim95==1));
totam96=length(find(undim96==1));
totam97=length(find(undim97==1));
totam98=length(find(undim98==1));
totam99=length(find(undim99==1));

tm9=totam91+totam92+totam93+totam94+totam95+totam96+totam97+totam98+totam99;

%10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
%Maximos=======================================
undis101=(mzds10<mzds1);
undis102=(mzds10<mzds2);
undis103=(mzds10<mzds3);
undis104=(mzds10<mzds4);
undis105=(mzds10<mzds5);
undis106=(mzds10<mzds6);
undis107=(mzds10<mzds7);
undis108=(mzds10<mzds8);
undis109=(mzds10<mzds9);
total101=length(find(undis101==1));
total102=length(find(undis102==1));
total103=length(find(undis103==1));
total104=length(find(undis104==1));
total105=length(find(undis105==1));
total106=length(find(undis106==1));
total107=length(find(undis107==1));
total108=length(find(undis108==1));
total109=length(find(undis109==1));
t10=total101+total102+total103+total104+total105+total106+total107+total108+total109;
%Minimos=======================================
undin101=(mzdn10<mzdn1);
undin102=(mzdn10<mzdn2);
```

```
undin103=(mzdn10<mzdn3);
undin104=(mzdn10<mzdn4);
undin105=(mzdn10<mzdn5);
undin106=(mzdn10<mzdn6);
undin107=(mzdn10<mzdn7);
undin108=(mzdn10<mzdn8);
undin109=(mzdn10<mzdn9);
totan101=length(find(undin101==1));
totan102=length(find(undin102==1));
totan103=length(find(undin103==1));
totan104=length(find(undin104==1));
totan105=length(find(undin105==1));
totan106=length(find(undin106==1));
totan107=length(find(undin107==1));
totan108=length(find(undin108==1));
totan109=length(find(undin109==1));

tn10=totan101+totan102+totan103+totan104+totan105+totan106+totan107+totan108+totan1
09;
%Moda=========================================
undim101=(mzdm10<mzdm1);
undim102=(mzdm10<mzdm2);
undim103=(mzdm10<mzdm3);
undim104=(mzdm10<mzdm4);
undim105=(mzdm10<mzdm5);
undim106=(mzdm10<mzdm6);
undim107=(mzdm10<mzdm7);
undim108=(mzdm10<mzdm8);
undim109=(mzdm10<mzdm9);
totam101=length(find(undim101==1));
totam102=length(find(undim102==1));
totam103=length(find(undim103==1));
totam104=length(find(undim104==1));
totam105=length(find(undim105==1));
totam106=length(find(undim106==1));
totam107=length(find(undim107==1));
totam108=length(find(undim108==1));
totam109=length(find(undim109==1));

tm10=totam101+totam102+totam103+totam104+totam105+totam106+totam107+totam108+t
otam109;

%-------------------------------------------
TT1=0;TT2=0;TT3=0;TT4=0;
TT5=0;TT6=0;TT7=0;TT8=0;
TT9=0;TT10=0;
%((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((

if (t1 > t2) && (t1 > t3) && (t1 > t4) && (t1 > t5) && (t1 > t6) && (t1 > t7) && (t1 > t8) &&
(t1 > t9) && (t1 > t10)
    plot(abs(fft(y0)));
```

```
        fprintf('1 AYUDA\n');
        TT1=1;
    end
    if (t2 > t1) && (t2 > t3) && (t2 > t4) && (t2 > t5) && (t2 > t6) && (t2 > t7) && (t2 > t8) &&
(t2 > t9) && (t2 > t10)
        plot(abs(fft(y1)));
        fprintf('1 BAÑO\n')
        TT2=1;
    end
    if (t3 > t1) && (t3 > t2) && (t3 > t4) && (t3 > t5) && (t3 > t6) && (t3 > t7) && (t3 > t8) &&
(t3 > t9) && (t3 > t10)
        plot(abs(fft(y2)));
        fprintf('1 CAMINO\n');
        TT3=1;
    end
    if (t4 > t1) && (t4 > t2) && (t4 > t3) && (t4 > t5) && (t4 > t6) && (t4 > t7) && (t4 > t8) &&
(t4 > t9) && (t4 > t10)
        plot(abs(fft(y3)));
        fprintf('1 CASA\n');
        TT4=1;
    end
    if (t5 > t1) && (t5 > t2) && (t5 > t3) && (t5 > t4) && (t5 > t6) && (t5 > t7) && (t5 > t8) &&
(t5 > t9) && (t5 > t10)
        plot(abs(fft(y4)));
        fprintf('1 COBIJA\n');
        TT5=1;
    end
    if (t6 > t1) && (t6 > t2) && (t6 > t3) && (t6 > t4) && (t6 > t5) && (t6 > t7) && (t6 > t8) &&
(t6 > t9) && (t6 > t10)
        plot(abs(fft(y5)));
        fprintf('1 COMER\n');
        TT6=1;
    end
    if (t7 > t1) && (t7 > t2) && (t7 > t3) && (t7 > t4) && (t7 > t5) && (t7 > t6) && (t7 > t8) &&
(t7 > t9) && (t7 > t10)
        plot(abs(fft(y6)));
        fprintf('1 DAME\n');
        TT7=1;
    end
    if (t8 > t1) && (t8 > t2) && (t8 > t3) && (t8 > t4) && (t8 > t5) && (t8 > t6) && (t8 > t7) &&
(t8 > t9) && (t8 > t10)
        plot(abs(fft(y7)));
        fprintf('1 ENFERMO\n');
        TT8=1;
    end
    if (t9 > t1) && (t9 > t2) && (t9 > t3) && (t9 > t4) && (t9 > t5) && (t9 > t6) && (t9 > t7) &&
(t9 > t8) && (t9 > t10)
        plot(abs(fft(y8)));
        fprintf('1 FRIO\n');
        TT9=1;
    end
```

```
    if (t10 > t1) && (t10 > t2) && (t10 > t3) && (t10 > t4) && (t10 > t5) && (t10 > t6) && (t10 >
t7) && (t10 > t8) && (t10 > t9)
        plot(abs(fft(y9)));
        fprintf('1 GRACIAS\n');
        TT10=1;
    end
    %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
    if (tn1 > tn2) && (tn1 > tn3) && (tn1 > tn4) && (tn1 > tn5) && (tn1 > tn6) && (tn1 > tn7)
&& (tn1 > tn8) && (tn1 > tn9) && (tn1 > tn10)
        plot(abs(fft(y0)));
        fprintf('2 AYUDA\n');
        TT1=1+TT1;
    end
    if (tn2 > tn1) && (tn2 > tn3) && (tn2 > tn4) && (tn2 > tn5) && (tn2 > tn6) && (tn2 > tn7)
&& (tn2 > tn8) && (tn2 > tn9) && (tn2 > tn10)
        plot(abs(fft(y1)));
        fprintf('2 BAÑO\n')
        TT2=1+TT2;
    end
    if (tn3 > tn1) && (tn3 > tn2) && (tn3 > tn4) && (tn3 > tn5) && (tn3 > tn6) && (tn3 > tn7)
&& (tn3 > tn8) && (tn3 > tn9) && (tn3 > tn10)
        plot(abs(fft(y2)));
        fprintf('2 CAMINO\n');
        TT3=1+TT3;
    end
    if (tn4 > tn1) && (tn4 > tn2) && (tn4 > tn3) && (tn4 > tn5) && (tn4 > tn6) && (tn4 > tn7)
&& (tn4 > tn8) && (tn4 > tn9) && (tn4 > tn10)
        plot(abs(fft(y3)));
        fprintf('2 CASA\n');
        TT4=1+TT4;
    end
    if (tn5 > tn1) && (tn5 > tn2) && (tn5 > tn3) && (tn5 > tn4) && (tn5 > tn6) && (tn5 > tn7)
&& (tn5 > tn8) && (tn5 > tn9) && (tn5 > tn10)
        plot(abs(fft(y4)));
        fprintf('2 COBIJA\n');
        TT5=1+TT5;
    end
    if (tn6 > tn1) && (tn6 > tn2) && (tn6 > tn3) && (tn6 > tn4) && (tn6 > tn5) && (tn6 > tn7)
&& (tn6 > tn8) && (tn6 > tn9) && (tn6 > tn10)
        plot(abs(fft(y5)));
        fprintf('2 COMER\n');
        TT6=1+TT6;
    end
    if (tn7 > tn1) && (tn7 > tn2) && (tn7 > tn3) && (tn7 > tn4) && (tn7 > tn5) && (tn7 > tn6)
&& (tn7 > tn8) && (tn7 > tn9) && (tn7 > tn10)
        plot(abs(fft(y6)));
        fprintf('2 DAME\n');
        TT7=1+TT7;
    end
    if (tn8 > tn1) && (tn8 > tn2) && (tn8 > tn3) && (tn8 > tn4) && (tn8 > tn5) && (tn8 > tn6)
&& (tn8 > tn7) && (tn8 > tn9) && (tn8 > tn10)
```

```
        plot(abs(fft(y7)));
        fprintf('2 ENFERMO\n');
        TT8=1+TT8;
    end
    if (tn9 > tn1) && (tn9 > tn2) && (tn9 > tn3) && (tn9 > tn4) && (tn9 > tn5) && (tn9 > tn6)
&& (tn9 > tn7) && (tn9 > tn8) && (tn9 > tn10)
        plot(abs(fft(y8)));
        fprintf('2 FRIO\n');
        TT9=1+TT9;
    end
    if (tn10 > tn1) && (tn10 > tn2) && (tn10 > tn3) && (tn10 > tn4) && (tn10 > tn5) && (tn10 >
tn6) && (tn10 > tn7) && (tn10 > tn8) && (tn10 > tn9)
        plot(abs(fft(y9)));
        fprintf('2 GRACIAS\n');
        TT10=1+TT10;
    end
    %))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
    if (tm1 > tm2) && (tm1 > tm3) && (tm1 > tm4) && (tm1 > tm5) && (tm1 > tm6) && (tm1 >
tm7) && (tm1 > tm8) && (tm1 > tm9) && (tm1 > tm10)
        plot(abs(fft(y0)));
        fprintf('3 AYUDA\n');
        TT1=1+TT1;
    end
    if (tm2 > tm1) && (tm2 > tm3) && (tm2 > tm4) && (tm2 > tm5) && (tm2 > tm6) && (tm2 >
tm7) && (tm2 > tm8) && (tm2 > tm9) && (tm2 > tm10)
        plot(abs(fft(y1)));
        fprintf('3 BAÑO\n')
        TT2=1+TT2;
    end
    if (tm3 > tm1) && (tm3 > tm2) && (tm3 > tm4) && (tm3 > tm5) && (tm3 > tm6) && (tm3 >
tm7) && (tm3 > tm8) && (tm3 > tm9) && (tm3 > tm10)
        plot(abs(fft(y2)));
        fprintf('3 CAMINO\n');
        TT3=1+TT3;
    end
    if (tm4 > tm1) && (tm4 > tm2) && (tm4 > tm3) && (tm4 > tm5) && (tm4 > tm6) && (tm4 >
tm7) && (tm4 > tm8) && (tm4 > tm9) && (tm4 > tm10)
        plot(abs(fft(y3)));
        fprintf('3 CASA\n');
        TT4=1+TT4;
    end
    if (tm5 > tm1) && (tm5 > tm2) && (tm5 > tm3) && (tm5 > tm4) && (tm5 > tm6) && (tm5 >
tm7) && (tm5 > tm8) && (tm5 > tm9) && (tm5 > tm10)
        plot(abs(fft(y4)));
        fprintf('3 COBIJA\n');
        TT5=1+TT5;
    end
    if (tm6 > tm1) && (tm6 > tm2) && (tm6 > tm3) && (tm6 > tm4) && (tm6 > tm5) && (tm6 >
tm7) && (tm6 > tm8) && (tm6 > tm9) && (tm6 > tm10)
        plot(abs(fft(y5)));
        fprintf('3 COMER\n');
```

```
        TT6=1+TT6;
    end
    if (tm7 > tm1) && (tm7 > tm2) && (tm7 > tm3) && (tm7 > tm4) && (tm7 > tm5) && (tm7 >
tm6) && (tm7 > tm8) && (tm7 > tm9) && (tm7 > tm10)
        plot(abs(fft(y6)));
        fprintf('3 DAME\n');
        TT7=1+TT7;
    end
    if (tm8 > tm1) && (tm8 > tm2) && (tm8 > tm3) && (tm8 > tm4) && (tm8 > tm5) && (tm8 >
tm6) && (tm8 > tm7) && (tm8 > tm9) && (tm8 > tm10)
        plot(abs(fft(y7)));
        fprintf('3 ENFERMO\n');
        TT8=1+TT8;
    end
    if (tm9 > tm1) && (tm9 > tm2) && (tm9 > tm3) && (tm9 > tm4) && (tm9 > tm5) && (tm9 >
tm6) && (tm9 > tm7) && (tm9 > tm8) && (tm9 > tm10)
        plot(abs(fft(y8)));
        fprintf('3 FRIO\n');
        TT9=1+TT9;
    end
    if (tm10 > tm1) && (tm10 > tm2) && (tm10 > tm3) && (tm10 > tm4) && (tm10 > tm5) &&
(tm10 > tm6) && (tm10 > tm7) && (tm10 > tm8) && (tm10 > tm9)
        plot(abs(fft(y9)));
        fprintf('3 GRACIAS\n');
        TT10=1+TT10;
    end
    %//////////////////////////////////////////////////////////////////////////////////
    if (te1 > te2) && (te1 > te3) && (te1 > te4) && (te1 > te5) && (te1 > te6) && (te1 > te7)
&& (te1 > te8) && (te1 > te9) && (te1 > te10)
        plot(abs(fft(y0)));
        fprintf('4 AYUDA\n');
        TT1=1+TT1;
    end
    if (te2 > te1) && (te2 > te3) && (te2 > te4) && (te2 > te5) && (te2 > te6) && (te2 > te7)
&& (te2 > te8) && (te2 > te9) && (te2 > te10)
        plot(abs(fft(y1)));
        fprintf('4 BAÑO\n')
        TT2=1+TT2;
    end
    if (te3 > te1) && (te3 > te2) && (te3 > te4) && (te3 > te5) && (te3 > te6) && (te3 > te7)
&& (te3 > te8) && (te3 > te9) && (te3 > te10)
        plot(abs(fft(y2)));
        fprintf('4 CAMINO\n');
        TT3=1+TT3;
    end
    if (te4 > te1) && (te4 > te2) && (te4 > te3) && (te4 > te5) && (te4 > te6) && (te4 > te7)
&& (te4 > te8) && (te4 > te9) && (te4 > te10)
        plot(abs(fft(y3)));
        fprintf('4 CASA\n');
        TT4=1+TT4;
    end
```

```
      if (te5 > te1) && (te5 > te2) && (te5 > te3) && (te5 > te4) && (te5 > te6) && (te5 > te7)
&& (te5 > te8) && (te5 > te9) && (te5 > te10)
          plot(abs(fft(y4)));
          fprintf('4 COBIJA\n');
          TT5=1+TT5;
      end
      if (te6 > te1) && (te6 > te2) && (te6 > te3) && (te6 > te4) && (te6 > te5) && (te6 > te7)
&& (te6 > te8) && (te6 > te9) && (te6 > te10)
          plot(abs(fft(y5)));
          fprintf('4 COMER\n');
          TT6=1+TT6;
      end
      if (te7 > te1) && (te7 > te2) && (te7 > te3) && (te7 > te4) && (te7 > te5) && (te7 > te6)
&& (te7 > te8) && (te7 > te9) && (te7 > te10)
          plot(abs(fft(y6)));
          fprintf('4 DAME\n');
          TT7=1+TT7;
      end
      if (te8 > te1) && (te8 > te2) && (te8 > te3) && (te8 > te4) && (te8 > te5) && (te8 > te6)
&& (te8 > te7) && (te8 > te9) && (te8 > te10)
          plot(abs(fft(y7)));
          fprintf('4 ENFERMO\n');
          TT8=1+TT8;
      end
      if (te9 > te1) && (te9 > te2) && (te9 > te3) && (te9 > te4) && (te9 > te5) && (te9 > te6)
&& (te9 > te7) && (te9 > te8) && (te9 > te10)
          plot(abs(fft(y8)));
          fprintf('4 FRIO\n');
          TT9=1+TT9;
      end
      if (te10 > te1) && (te10 > te2) && (te10 > te3) && (te10 > te4) && (te10 > te5) && (te10 >
te6) && (te10 > te7) && (te10 > te8) && (te10 > te9)
          plot(abs(fft(y9)));
          fprintf('4 GRACIAS\n');
          TT10=1+TT10;
      end

%=================================================================
===
      if TT1>=2
          msgbox('Identificado: AYUDA','1','warn');
          [v fs]=wavread('ayuda');
          soundsc(v,fs)
      end
      if TT2>=2
          msgbox('Identificado: BAÑO','1','warn');
          [v fs]=wavread('baño');
          soundsc(v,fs)
      end
      if TT3>=2
          msgbox('Identificado: CAMINO','1','warn');
```

```
        [v fs]=wavread('camino');
        soundsc(v,fs)
    end
    if TT4>=2
        msgbox('Identificado: CASA','1','warn');
        [v fs]=wavread('casa');
        soundsc(v,fs)
    end
    if TT5>=2
        msgbox('Identificado: COBIJA','1','warn');
        [v fs]=wavread('cobija');
        soundsc(v,fs)
    end
    if TT6>=2
        msgbox('Identificado: COMER','1','warn');
        [v fs]=wavread('comer');
        soundsc(v,fs)
    end
    if TT7>=2
        msgbox('Identificado: DAME','1','warn');
        [v fs]=wavread('dame');
        soundsc(v,fs)
    end
    if TT8>=2
        msgbox('Identificado: ENFERMO','1','warn');
        [v fs]=wavread('enfermo');
        soundsc(v,fs)
    end
    if TT9>=2
        msgbox('Identificado: FRIO','1','warn');
        [v fs]=wavread('frio');
        soundsc(v,fs)
    end
    if TT10>=2
        msgbox('Identificado: GRACIAS','1','warn');
        [v fs]=wavread('gracias');
        soundsc(v,fs)
    end
    % ========================================
case 2
    d0=distancia(f0,rec1);
    d1=distancia(f1,rec1);
    d2=distancia(f2,rec1);
    d3=distancia(f3,rec1);
    d4=distancia(f4,rec1);
    d5=distancia(f5,rec1);
    d6=distancia(f6,rec1);
    d7=distancia(f7,rec1);
    d8=distancia(f8,rec1);
    d9=distancia(f9,rec1);
```

```
Ds1=size(d1);Ds2=size(d2);
Ds3=size(d3);Ds4=size(d4);
Ds5=size(d5);Ds6=size(d6);
Ds7=size(d7);Ds8=size(d8);
Ds9=size(d9);Ds0=size(d0);

dista1=cat(2,Ds1(1),Ds2(1),Ds3(1),Ds4(1),Ds5(1),Ds6(1),Ds7(1),Ds8(1),Ds9(1),Ds0(1));
dista2=cat(2,Ds1(2),Ds2(2),Ds3(2),Ds4(2),Ds5(2),Ds6(2),Ds7(2),Ds8(2),Ds9(2),Ds0(2));

minds1=min(dista1);
minds2=min(dista2);

d11=d1(1:minds1,1:minds2);
d12=d2(1:minds1,1:minds2);
d13=d3(1:minds1,1:minds2);
d14=d4(1:minds1,1:minds2);
d15=d5(1:minds1,1:minds2);
d16=d6(1:minds1,1:minds2);
d17=d7(1:minds1,1:minds2);
d18=d8(1:minds1,1:minds2);
d19=d9(1:minds1,1:minds2);
d20=d0(1:minds1,1:minds2);

%MAXIMOS.....
mzds1=max(d11);
mzds2=max(d12);
mzds3=max(d13);
mzds4=max(d14);
mzds5=max(d15);
mzds6=max(d16);
mzds7=max(d17);
mzds8=max(d18);
mzds9=max(d19);
mzds10=max(d20);

%MIMINOS.....
mzdn1=min(d11);
mzdn2=min(d12);
mzdn3=min(d13);
mzdn4=min(d14);
mzdn5=min(d15);
mzdn6=min(d16);
mzdn7=min(d17);
mzdn8=min(d18);
mzdn9=min(d19);
mzdn10=min(d20);
%MODA.....
mzdm1=mode(d11);
mzdm2=mode(d12);
mzdm3=mode(d13);
mzdm4=mode(d14);
```

```
mzdm5=mode(d15);
mzdm6=mode(d16);
mzdm7=mode(d17);
mzdm8=mode(d18);
mzdm9=mode(d19);
mzdm10=mode(d20);
%1111111111111111111111111111111111111111111111111111
%Maximos=======================================
undis11=(mzds1<mzds2);
undis12=(mzds1<mzds3);
undis13=(mzds1<mzds4);
undis14=(mzds1<mzds5);
undis15=(mzds1<mzds6);
undis16=(mzds1<mzds7);
undis17=(mzds1<mzds8);
undis18=(mzds1<mzds9);
undis19=(mzds1<mzds10);
total11=length(find(undis11==1));
total12=length(find(undis12==1));
total13=length(find(undis13==1));
total14=length(find(undis14==1));
total15=length(find(undis15==1));
total16=length(find(undis16==1));
total17=length(find(undis17==1));
total18=length(find(undis18==1));
total19=length(find(undis19==1));
t1=total11+total12+total13+total14+total15+total16+total17+total18+total19;
%Minimos=======================================
undin11=(mzdn1<mzdn2);
undin12=(mzdn1<mzdn3);
undin13=(mzdn1<mzdn4);
undin14=(mzdn1<mzdn5);
undin15=(mzdn1<mzdn6);
undin16=(mzdn1<mzdn7);
undin17=(mzdn1<mzdn8);
undin18=(mzdn1<mzdn9);
undin19=(mzdn1<mzdn10);
totan11=length(find(undin11==1));
totan12=length(find(undin12==1));
totan13=length(find(undin13==1));
totan14=length(find(undin14==1));
totan15=length(find(undin15==1));
totan16=length(find(undin16==1));
totan17=length(find(undin17==1));
totan18=length(find(undin18==1));
totan19=length(find(undin19==1));
tn1=totan11+totan12+totan13+totan14+totan15+totan16+totan17+totan18+totan19;
%Moda=======================================
undim11=(mzdm1<mzdm2);
undim12=(mzdm1<mzdm3);
undim13=(mzdm1<mzdm4);
```

```
        undim14=(mzdm1<mzdm5);
        undim15=(mzdm1<mzdm6);
        undim16=(mzdm1<mzdm7);
        undim17=(mzdm1<mzdm8);
        undim18=(mzdm1<mzdm9);
        undim19=(mzdm1<mzdm10);
        totam11=length(find(undim11==1));
        totam12=length(find(undim12==1));
        totam13=length(find(undim13==1));
        totam14=length(find(undim14==1));
        totam15=length(find(undim15==1));
        totam16=length(find(undim16==1));
        totam17=length(find(undim17==1));
        totam18=length(find(undim18==1));
        totam19=length(find(undim19==1));

    tm1=totam11+totam12+totam13+totam14+totam15+totam16+totam17+totam18+totam19;
        % 22222222222222222222222222222222222222222222222222
        %Maximos=======================================
        undis21=(mzds2<mzds1);
        undis22=(mzds2<mzds3);
        undis23=(mzds2<mzds4);
        undis24=(mzds2<mzds5);
        undis25=(mzds2<mzds6);
        undis26=(mzds2<mzds7);
        undis27=(mzds2<mzds8);
        undis28=(mzds2<mzds9);
        undis29=(mzds2<mzds10);
        total21=length(find(undis21==1));
        total22=length(find(undis22==1));
        total23=length(find(undis23==1));
        total24=length(find(undis24==1));
        total25=length(find(undis25==1));
        total26=length(find(undis26==1));
        total27=length(find(undis27==1));
        total28=length(find(undis28==1));
        total29=length(find(undis29==1));
        t2=total21+total22+total23+total24+total25+total26+total27+total28+total29;
        %Minimos=======================================
        undin21=(mzdn2<mzdn1);
        undin22=(mzdn2<mzdn3);
        undin23=(mzdn2<mzdn4);
        undin24=(mzdn2<mzdn5);
        undin25=(mzdn2<mzdn6);
        undin26=(mzdn2<mzdn7);
        undin27=(mzdn2<mzdn8);
        undin28=(mzdn2<mzdn9);
        undin29=(mzdn2<mzdn10);
        totan21=length(find(undin21==1));
        totan22=length(find(undin22==1));
        totan23=length(find(undin23==1));
```

```
totan24=length(find(undin24==1));
totan25=length(find(undin25==1));
totan26=length(find(undin26==1));
totan27=length(find(undin27==1));
totan28=length(find(undin28==1));
totan29=length(find(undin29==1));
tn2=totan21+totan22+totan23+totan24+totan25+totan26+totan27+totan28+totan29;
%Moda=======================================
undim21=(mzdm2<mzdm1);
undim22=(mzdm2<mzdm3);
undim23=(mzdm2<mzdm4);
undim24=(mzdm2<mzdm5);
undim25=(mzdm2<mzdm6);
undim26=(mzdm2<mzdm7);
undim27=(mzdm2<mzdm8);
undim28=(mzdm2<mzdm9);
undim29=(mzdm2<mzdm10);
totam21=length(find(undim21==1));
totam22=length(find(undim22==1));
totam23=length(find(undim23==1));
totam24=length(find(undim24==1));
totam25=length(find(undim25==1));
totam26=length(find(undim26==1));
totam27=length(find(undim27==1));
totam28=length(find(undim28==1));
totam29=length(find(undim29==1));

tm2=totam21+totam22+totam23+totam24+totam25+totam26+totam27+totam28+totam29;
%3333333333333333333333333333333333333333333333333333333
%Maximos=======================================
undis31=(mzds3<mzds1);
undis32=(mzds3<mzds2);
undis33=(mzds3<mzds4);
undis34=(mzds3<mzds5);
undis35=(mzds3<mzds6);
undis36=(mzds3<mzds7);
undis37=(mzds3<mzds8);
undis38=(mzds3<mzds9);
undis39=(mzds3<mzds10);
total31=length(find(undis31==1));
total32=length(find(undis32==1));
total33=length(find(undis33==1));
total34=length(find(undis34==1));
total35=length(find(undis35==1));
total36=length(find(undis36==1));
total37=length(find(undis37==1));
total38=length(find(undis38==1));
total39=length(find(undis39==1));
t3=total31+total32+total33+total34+total35+total36+total37+total38+total39;
%Minimos=======================================
undin31=(mzdn3<mzdn1);
```

```
undin32=(mzdn3<mzdn2);
undin33=(mzdn3<mzdn4);
undin34=(mzdn3<mzdn5);
undin35=(mzdn3<mzdn6);
undin36=(mzdn3<mzdn7);
undin37=(mzdn3<mzdn8);
undin38=(mzdn3<mzdn9);
undin39=(mzdn3<mzdn10);
totan31=length(find(undin31==1));
totan32=length(find(undin32==1));
totan33=length(find(undin33==1));
totan34=length(find(undin34==1));
totan35=length(find(undin35==1));
totan36=length(find(undin36==1));
totan37=length(find(undin37==1));
totan38=length(find(undin38==1));
totan39=length(find(undin39==1));
tn3=totan31+totan32+totan33+totan34+totan35+totan36+totan37+totan38+totan39;
%Moda=========================================
undim31=(mzdm3<mzdm1);
undim32=(mzdm3<mzdm2);
undim33=(mzdm3<mzdm4);
undim34=(mzdm3<mzdm5);
undim35=(mzdm3<mzdm6);
undim36=(mzdm3<mzdm7);
undim37=(mzdm3<mzdm8);
undim38=(mzdm3<mzdm9);
undim39=(mzdm3<mzdm10);
totam31=length(find(undim31==1));
totam32=length(find(undim32==1));
totam33=length(find(undim33==1));
totam34=length(find(undim34==1));
totam35=length(find(undim35==1));
totam36=length(find(undim36==1));
totam37=length(find(undim37==1));
totam38=length(find(undim38==1));
totam39=length(find(undim39==1));

tm3=totam31+totam32+totam33+totam34+totam35+totam36+totam37+totam38+totam39;
%4444444444444444444444444444444444444444444444444444
%Maximos=========================================
undis41=(mzds4<mzds1);
undis42=(mzds4<mzds2);
undis43=(mzds4<mzds3);
undis44=(mzds4<mzds5);
undis45=(mzds4<mzds6);
undis46=(mzds4<mzds7);
undis47=(mzds4<mzds8);
undis48=(mzds4<mzds9);
undis49=(mzds4<mzds10);
total41=length(find(undis41==1));
```

```
total42=length(find(undis42==1));
total43=length(find(undis43==1));
total44=length(find(undis44==1));
total45=length(find(undis45==1));
total46=length(find(undis46==1));
total47=length(find(undis47==1));
total48=length(find(undis48==1));
total49=length(find(undis49==1));
t4=total41+total42+total43+total44+total45+total46+total47+total48+total49;
%Minimos=======================================
undin41=(mzdn4<mzdn1);
undin42=(mzdn4<mzdn2);
undin43=(mzdn4<mzdn3);
undin44=(mzdn4<mzdn5);
undin45=(mzdn4<mzdn6);
undin46=(mzdn4<mzdn7);
undin47=(mzdn4<mzdn8);
undin48=(mzdn4<mzdn9);
undin49=(mzdn4<mzdn10);
totan41=length(find(undin41==1));
totan42=length(find(undin42==1));
totan43=length(find(undin43==1));
totan44=length(find(undin44==1));
totan45=length(find(undin45==1));
totan46=length(find(undin46==1));
totan47=length(find(undin47==1));
totan48=length(find(undin48==1));
totan49=length(find(undin49==1));
tn4=totan41+totan42+totan43+totan44+totan45+totan46+totan47+totan48+totan49;
%Moda=======================================
undim41=(mzdm4<mzdm1);
undim42=(mzdm4<mzdm2);
undim43=(mzdm4<mzdm3);
undim44=(mzdm4<mzdm5);
undim45=(mzdm4<mzdm6);
undim46=(mzdm4<mzdm7);
undim47=(mzdm4<mzdm8);
undim48=(mzdm4<mzdm9);
undim49=(mzdm4<mzdm10);
totam41=length(find(undim41==1));
totam42=length(find(undim42==1));
totam43=length(find(undim43==1));
totam44=length(find(undim44==1));
totam45=length(find(undim45==1));
totam46=length(find(undim46==1));
totam47=length(find(undim47==1));
totam48=length(find(undim48==1));
totam49=length(find(undim49==1));

tm4=totam41+totam42+totam43+totam44+totam45+totam46+totam47+totam48+totam49;
```

```
%5555555555555555555555555555555555555555555555555555555555
    %Maximos=======================================
    undis51=(mzds5<mzds1);
    undis52=(mzds5<mzds2);
    undis53=(mzds5<mzds3);
    undis54=(mzds5<mzds4);
    undis55=(mzds5<mzds6);
    undis56=(mzds5<mzds7);
    undis57=(mzds5<mzds8);
    undis58=(mzds5<mzds9);
    undis59=(mzds5<mzds10);
    total51=length(find(undis51==1));
    total52=length(find(undis52==1));
    total53=length(find(undis53==1));
    total54=length(find(undis54==1));
    total55=length(find(undis55==1));
    total56=length(find(undis56==1));
    total57=length(find(undis57==1));
    total58=length(find(undis58==1));
    total59=length(find(undis59==1));
    t5=total51+total52+total53+total54+total55+total56+total57+total58+total59;
    %Minimos=======================================
    undin51=(mzdn5<mzdn1);
    undin52=(mzdn5<mzdn2);
    undin53=(mzdn5<mzdn3);
    undin54=(mzdn5<mzdn4);
    undin55=(mzdn5<mzdn6);
    undin56=(mzdn5<mzdn7);
    undin57=(mzdn5<mzdn8);
    undin58=(mzdn5<mzdn9);
    undin59=(mzdn5<mzdn10);
    totan51=length(find(undin51==1));
    totan52=length(find(undin52==1));
    totan53=length(find(undin53==1));
    totan54=length(find(undin54==1));
    totan55=length(find(undin55==1));
    totan56=length(find(undin56==1));
    totan57=length(find(undin57==1));
    totan58=length(find(undin58==1));
    totan59=length(find(undin59==1));
    tn5=totan51+totan52+totan53+totan54+totan55+totan56+totan57+totan58+totan59;
    %Moda===========================================
    undim51=(mzdm5<mzdm1);
    undim52=(mzdm5<mzdm2);
    undim53=(mzdm5<mzdm3);
    undim54=(mzdm5<mzdm4);
    undim55=(mzdm5<mzdm6);
    undim56=(mzdm5<mzdm7);
    undim57=(mzdm5<mzdm8);
    undim58=(mzdm5<mzdm9);
    undim59=(mzdm5<mzdm10);
```

```
        totam51=length(find(undim51==1));
        totam52=length(find(undim52==1));
        totam53=length(find(undim53==1));
        totam54=length(find(undim54==1));
        totam55=length(find(undim55==1));
        totam56=length(find(undim56==1));
        totam57=length(find(undim57==1));
        totam58=length(find(undim58==1));
        totam59=length(find(undim59==1));

tm5=totam51+totam52+totam53+totam54+totam55+totam56+totam57+totam58+totam59;
%6666666666666666666666666666666666666666666666666666666666666666666666
        %Maximos=========================================
        undis61=(mzds6<mzds1);
        undis62=(mzds6<mzds2);
        undis63=(mzds6<mzds3);
        undis64=(mzds6<mzds4);
        undis65=(mzds6<mzds5);
        undis66=(mzds6<mzds7);
        undis67=(mzds6<mzds8);
        undis68=(mzds6<mzds9);
        undis69=(mzds6<mzds10);
        total61=length(find(undis61==1));
        total62=length(find(undis62==1));
        total63=length(find(undis63==1));
        total64=length(find(undis64==1));
        total65=length(find(undis65==1));
        total66=length(find(undis66==1));
        total67=length(find(undis67==1));
        total68=length(find(undis68==1));
        total69=length(find(undis69==1));
        t6=total61+total62+total63+total64+total65+total66+total67+total68+total69;
        %Minimos=========================================
        undin61=(mzdn6<mzdn1);
        undin62=(mzdn6<mzdn2);
        undin63=(mzdn6<mzdn3);
        undin64=(mzdn6<mzdn4);
        undin65=(mzdn6<mzdn5);
        undin66=(mzdn6<mzdn7);
        undin67=(mzdn6<mzdn8);
        undin68=(mzdn6<mzdn9);
        undin69=(mzdn6<mzdn10);
        totan61=length(find(undin61==1));
        totan62=length(find(undin62==1));
        totan63=length(find(undin63==1));
        totan64=length(find(undin64==1));
        totan65=length(find(undin65==1));
        totan66=length(find(undin66==1));
        totan67=length(find(undin67==1));
        totan68=length(find(undin68==1));
        totan69=length(find(undin69==1));
```

```
tn6=totan61+totan62+totan63+totan64+totan65+totan66+totan67+totan68+totan69;
%Moda=======================================
undim61=(mzdm6<mzdm1);
undim62=(mzdm6<mzdm2);
undim63=(mzdm6<mzdm3);
undim64=(mzdm6<mzdm4);
undim65=(mzdm6<mzdm5);
undim66=(mzdm6<mzdm7);
undim67=(mzdm6<mzdm8);
undim68=(mzdm6<mzdm9);
undim69=(mzdm6<mzdm10);
totam61=length(find(undim61==1));
totam62=length(find(undim62==1));
totam63=length(find(undim63==1));
totam64=length(find(undim64==1));
totam65=length(find(undim65==1));
totam66=length(find(undim66==1));
totam67=length(find(undim67==1));
totam68=length(find(undim68==1));
totam69=length(find(undim69==1));

tm6=totam61+totam62+totam63+totam64+totam65+totam66+totam67+totam68+totam69;
%77777777777777777777777777777777777777777777777777777777777777777777
%Maximos=======================================
undis71=(mzds7<mzds1);
undis72=(mzds7<mzds2);
undis73=(mzds7<mzds3);
undis74=(mzds7<mzds4);
undis75=(mzds7<mzds5);
undis76=(mzds7<mzds6);
undis77=(mzds7<mzds8);
undis78=(mzds7<mzds9);
undis79=(mzds7<mzds10);
total71=length(find(undis71==1));
total72=length(find(undis72==1));
total73=length(find(undis73==1));
total74=length(find(undis74==1));
total75=length(find(undis75==1));
total76=length(find(undis76==1));
total77=length(find(undis77==1));
total78=length(find(undis78==1));
total79=length(find(undis79==1));
t7=total71+total72+total73+total74+total75+total76+total77+total78+total79;
%Minimos=======================================
undin71=(mzdn7<mzdn1);
undin72=(mzdn7<mzdn2);
undin73=(mzdn7<mzdn3);
undin74=(mzdn7<mzdn4);
undin75=(mzdn7<mzdn5);
undin76=(mzdn7<mzdn6);
undin77=(mzdn7<mzdn8);
```

```
undin78=(mzdn7<mzdn9);
undin79=(mzdn7<mzdn10);
totan71=length(find(undin71==1));
totan72=length(find(undin72==1));
totan73=length(find(undin73==1));
totan74=length(find(undin74==1));
totan75=length(find(undin75==1));
totan76=length(find(undin76==1));
totan77=length(find(undin77==1));
totan78=length(find(undin78==1));
totan79=length(find(undin79==1));
tn7=totan71+totan72+totan73+totan74+totan75+totan76+totan77+totan78+totan79;
%Moda=========================================
undim71=(mzdm7<mzdm1);
undim72=(mzdm7<mzdm2);
undim73=(mzdm7<mzdm3);
undim74=(mzdm7<mzdm4);
undim75=(mzdm7<mzdm5);
undim76=(mzdm7<mzdm6);
undim77=(mzdm7<mzdm8);
undim78=(mzdm7<mzdm9);
undim79=(mzdm7<mzdm10);
totam71=length(find(undim71==1));
totam72=length(find(undim72==1));
totam73=length(find(undim73==1));
totam74=length(find(undim74==1));
totam75=length(find(undim75==1));
totam76=length(find(undim76==1));
totam77=length(find(undim77==1));
totam78=length(find(undim78==1));
totam79=length(find(undim79==1));

tm7=totam71+totam72+totam73+totam74+totam75+totam76+totam77+totam78+totam79;
%888888888888888888888888888888888888888888888888888888888888
%Maximos=======================================
undis81=(mzds8<mzds1);
undis82=(mzds8<mzds2);
undis83=(mzds8<mzds3);
undis84=(mzds8<mzds4);
undis85=(mzds8<mzds5);
undis86=(mzds8<mzds6);
undis87=(mzds8<mzds7);
undis88=(mzds8<mzds9);
undis89=(mzds8<mzds10);
total81=length(find(undis81==1));
total82=length(find(undis82==1));
total83=length(find(undis83==1));
total84=length(find(undis84==1));
total85=length(find(undis85==1));
total86=length(find(undis86==1));
total87=length(find(undis87==1));
```

```
total88=length(find(undis88==1));
total89=length(find(undis89==1));
t8=total81+total82+total83+total84+total85+total86+total87+total88+total89;
%Minimos=======================================
undin81=(mzdn8<mzdn1);
undin82=(mzdn8<mzdn2);
undin83=(mzdn8<mzdn3);
undin84=(mzdn8<mzdn4);
undin85=(mzdn8<mzdn5);
undin86=(mzdn8<mzdn6);
undin87=(mzdn8<mzdn7);
undin88=(mzdn8<mzdn9);
undin89=(mzdn8<mzdn10);
totan81=length(find(undin81==1));
totan82=length(find(undin82==1));
totan83=length(find(undin83==1));
totan84=length(find(undin84==1));
totan85=length(find(undin85==1));
totan86=length(find(undin86==1));
totan87=length(find(undin87==1));
totan88=length(find(undin88==1));
totan89=length(find(undin89==1));
tn8=totan81+totan82+totan83+totan84+totan85+totan86+totan87+totan88+totan89;
%Moda=======================================
undim81=(mzdm8<mzdm1);
undim82=(mzdm8<mzdm2);
undim83=(mzdm8<mzdm3);
undim84=(mzdm8<mzdm4);
undim85=(mzdm8<mzdm5);
undim86=(mzdm8<mzdm6);
undim87=(mzdm8<mzdm7);
undim88=(mzdm8<mzdm9);
undim89=(mzdm8<mzdm10);
totam81=length(find(undim81==1));
totam82=length(find(undim82==1));
totam83=length(find(undim83==1));
totam84=length(find(undim84==1));
totam85=length(find(undim85==1));
totam86=length(find(undim86==1));
totam87=length(find(undim87==1));
totam88=length(find(undim88==1));
totam89=length(find(undim89==1));

tm8=totam81+totam82+totam83+totam84+totam85+totam86+totam87+totam88+totam89;
%999999999999999999999999999999999999999999999999999999999999
%Maximos=======================================
undis91=(mzds9<mzds1);
undis92=(mzds9<mzds2);
undis93=(mzds9<mzds3);
undis94=(mzds9<mzds4);
undis95=(mzds9<mzds5);
```

```
undis96=(mzds9<mzds6);
undis97=(mzds9<mzds7);
undis98=(mzds9<mzds8);
undis99=(mzds9<mzds10);
total91=length(find(undis91==1));
total92=length(find(undis92==1));
total93=length(find(undis93==1));
total94=length(find(undis94==1));
total95=length(find(undis95==1));
total96=length(find(undis96==1));
total97=length(find(undis97==1));
total98=length(find(undis98==1));
total99=length(find(undis99==1));
t9=total91+total92+total93+total94+total95+total96+total97+total98+total99;
%Minimos=======================================
undin91=(mzdn9<mzdn1);
undin92=(mzdn9<mzdn2);
undin93=(mzdn9<mzdn3);
undin94=(mzdn9<mzdn4);
undin95=(mzdn9<mzdn5);
undin96=(mzdn9<mzdn6);
undin97=(mzdn9<mzdn7);
undin98=(mzdn9<mzdn8);
undin99=(mzdn9<mzdn10);
totan91=length(find(undin91==1));
totan92=length(find(undin92==1));
totan93=length(find(undin93==1));
totan94=length(find(undin94==1));
totan95=length(find(undin95==1));
totan96=length(find(undin96==1));
totan97=length(find(undin97==1));
totan98=length(find(undin98==1));
totan99=length(find(undin99==1));
tn9=totan91+totan92+totan93+totan94+totan95+totan96+totan97+totan98+totan99;
%Moda==========================================
undim91=(mzdm9<mzdm1);
undim92=(mzdm9<mzdm2);
undim93=(mzdm9<mzdm3);
undim94=(mzdm9<mzdm4);
undim95=(mzdm9<mzdm5);
undim96=(mzdm9<mzdm6);
undim97=(mzdm9<mzdm7);
undim98=(mzdm9<mzdm8);
undim99=(mzdm9<mzdm10);
totam91=length(find(undim91==1));
totam92=length(find(undim92==1));
totam93=length(find(undim93==1));
totam94=length(find(undim94==1));
totam95=length(find(undim95==1));
totam96=length(find(undim96==1));
totam97=length(find(undim97==1));
```

```
totam98=length(find(undim98==1));
totam99=length(find(undim99==1));

tm9=totam91+totam92+totam93+totam94+totam95+totam96+totam97+totam98+totam99;
%10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
%Maximos=======================================
undis101=(mzds10<mzds1);
undis102=(mzds10<mzds2);
undis103=(mzds10<mzds3);
undis104=(mzds10<mzds4);
undis105=(mzds10<mzds5);
undis106=(mzds10<mzds6);
undis107=(mzds10<mzds7);
undis108=(mzds10<mzds8);
undis109=(mzds10<mzds9);
total101=length(find(undis101==1));
total102=length(find(undis102==1));
total103=length(find(undis103==1));
total104=length(find(undis104==1));
total105=length(find(undis105==1));
total106=length(find(undis106==1));
total107=length(find(undis107==1));
total108=length(find(undis108==1));
total109=length(find(undis109==1));
t10=total101+total102+total103+total104+total105+total106+total107+total108+total109;
%Minimos=======================================
undin101=(mzdn10<mzdn1);
undin102=(mzdn10<mzdn2);
undin103=(mzdn10<mzdn3);
undin104=(mzdn10<mzdn4);
undin105=(mzdn10<mzdn5);
undin106=(mzdn10<mzdn6);
undin107=(mzdn10<mzdn7);
undin108=(mzdn10<mzdn8);
undin109=(mzdn10<mzdn9);
totan101=length(find(undin101==1));
totan102=length(find(undin102==1));
totan103=length(find(undin103==1));
totan104=length(find(undin104==1));
totan105=length(find(undin105==1));
totan106=length(find(undin106==1));
totan107=length(find(undin107==1));
totan108=length(find(undin108==1));
totan109=length(find(undin109==1));

tn10=totan101+totan102+totan103+totan104+totan105+totan106+totan107+totan108+totan1
09;
%Moda=======================================
undim101=(mzdm10<mzdm1);
undim102=(mzdm10<mzdm2);
undim103=(mzdm10<mzdm3);
```

```
        undim104=(mzdm10<mzdm4);
        undim105=(mzdm10<mzdm5);
        undim106=(mzdm10<mzdm6);
        undim107=(mzdm10<mzdm7);
        undim108=(mzdm10<mzdm8);
        undim109=(mzdm10<mzdm9);
        totam101=length(find(undim101==1));
        totam102=length(find(undim102==1));
        totam103=length(find(undim103==1));
        totam104=length(find(undim104==1));
        totam105=length(find(undim105==1));
        totam106=length(find(undim106==1));
        totam107=length(find(undim107==1));
        totam108=length(find(undim108==1));
        totam109=length(find(undim109==1));

tm10=totam101+totam102+totam103+totam104+totam105+totam106+totam107+totam108+t
otam109;
        %-------------------------------------------
        TT1=0;TT2=0;TT3=0;TT4=0;
        TT5=0;TT6=0;TT7=0;TT8=0;
        TT9=0;TT10=0;
        %((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((

        if (t1 > t2) && (t1 > t3) && (t1 > t4) && (t1 > t5) && (t1 > t6) && (t1 > t7) && (t1 > t8) &&
(t1 > t9) && (t1 > t10)
           plot(abs(fft(y0)));
           fprintf('1 HAMBRE\n');
           TT1=1;
        end
        if (t2 > t1) && (t2 > t3) && (t2 > t4) && (t2 > t5) && (t2 > t6) && (t2 > t7) && (t2 > t8) &&
(t2 > t9) && (t2 > t10)
           plot(abs(fft(y1)));
           fprintf('1 HIJO\n');
           TT2=1;
        end
        if (t3 > t1) && (t3 > t2) && (t3 > t4) && (t3 > t5) && (t3 > t6) && (t3 > t7) && (t3 > t8) &&
(t3 > t9) && (t3 > t10)
           plot(abs(fft(y2)));
           fprintf('1 MARIA\n');
           TT3=1;
        end
        if (t4 > t1) && (t4 > t2) && (t4 > t3) && (t4 > t5) && (t4 > t6) && (t4 > t7) && (t4 > t8) &&
(t4 > t9) && (t4 > t10)
           plot(abs(fft(y3)));
           fprintf('1 QUIERO\n');
           TT4=1;
        end
        if (t5 > t1) && (t5 > t2) && (t5 > t3) && (t5 > t4) && (t5 > t6) && (t5 > t7) && (t5 > t8) &&
(t5 > t9) && (t5 > t10)
           plot(abs(fft(y4)));
```

```
        fprintf('1 ROPA\n');
        TT5=1;
    end
    if (t6 > t1) && (t6 > t2) && (t6 > t3) && (t6 > t4) && (t6 > t5) && (t6 > t7) && (t6 > t8) &&
(t6 > t9) && (t6 > t10)
        plot(abs(fft(y5)));
        fprintf('1 SED\n');
        TT6=1;
    end
    if (t7 > t1) && (t7 > t2) && (t7 > t3) && (t7 > t4) && (t7 > t5) && (t7 > t6) && (t7 > t8) &&
(t7 > t9) && (t7 > t10)
        plot(abs(fft(y6)));
        fprintf('1 TENGO\n');
        TT7=1;
    end
    if (t8 > t1) && (t8 > t2) && (t8 > t3) && (t8 > t4) && (t8 > t5) && (t8 > t6) && (t8 > t7) &&
(t8 > t9) && (t8 > t10)
        plot(abs(fft(y7)));
        fprintf('1 VEN\n');
        TT8=1;
    end
    if (t9 > t1) && (t9 > t2) && (t9 > t3) && (t9 > t4) && (t9 > t5) && (t9 > t6) && (t9 > t7) &&
(t9 > t8) && (t9 > t10)
        plot(abs(fft(y8)));
        fprintf('1 YO\n');
        TT9=1;
    end
    if (t10 > t1) && (t10 > t2) && (t10 > t3) && (t10 > t4) && (t10 > t5) && (t10 > t6) && (t10 >
t7) && (t10 > t8) && (t10 > t9)
        plot(abs(fft(y9)));
        fprintf('1 ZAPATO\n');
        TT10=1;
    end
    %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
    if (tn1 > tn2) && (tn1 > tn3) && (tn1 > tn4) && (tn1 > tn5) && (tn1 > tn6) && (tn1 > tn7)
&& (tn1 > tn8) && (tn1 > tn9) && (tn1 > tn10)
        plot(abs(fft(y0)));
        fprintf('2 HAMBRE\n');
        TT1=1+TT1;
    end
    if (tn2 > tn1) && (tn2 > tn3) && (tn2 > tn4) && (tn2 > tn5) && (tn2 > tn6) && (tn2 > tn7)
&& (tn2 > tn8) && (tn2 > tn9) && (tn2 > tn10)
        plot(abs(fft(y1)));
        fprintf('2 HIJO\n')
        TT2=1+TT2;
    end
    if (tn3 > tn1) && (tn3 > tn2) && (tn3 > tn4) && (tn3 > tn5) && (tn3 > tn6) && (tn3 > tn7)
&& (tn3 > tn8) && (tn3 > tn9) && (tn3 > tn10)
        plot(abs(fft(y2)));
        fprintf('2 MARIA\n');
        TT3=1+TT3;
```

```
    end
    if (tn4 > tn1) && (tn4 > tn2) && (tn4 > tn3) && (tn4 > tn5) && (tn4 > tn6) && (tn4 > tn7)
&& (tn4 > tn8) && (tn4 > tn9) && (tn4 > tn10)
        plot(abs(fft(y3)));
        fprintf('2 QUIERO\n');
        TT4=1+TT4;
    end
    if (tn5 > tn1) && (tn5 > tn2) && (tn5 > tn3) && (tn5 > tn4) && (tn5 > tn6) && (tn5 > tn7)
&& (tn5 > tn8) && (tn5 > tn9) && (tn5 > tn10)
        plot(abs(fft(y4)));
        fprintf('2 ROPA\n');
        TT5=1+TT5;
    end
    if (tn6 > tn1) && (tn6 > tn2) && (tn6 > tn3) && (tn6 > tn4) && (tn6 > tn5) && (tn6 > tn7)
&& (tn6 > tn8) && (tn6 > tn9) && (tn6 > tn10)
        plot(abs(fft(y5)));
        fprintf('2 SED\n');
        TT6=1+TT6;
    end
    if (tn7 > tn1) && (tn7 > tn2) && (tn7 > tn3) && (tn7 > tn4) && (tn7 > tn5) && (tn7 > tn6)
&& (tn7 > tn8) && (tn7 > tn9) && (tn7 > tn10)
        plot(abs(fft(y6)));
        fprintf('2 TENGO\n');
        TT7=1+TT7;
    end
    if (tn8 > tn1) && (tn8 > tn2) && (tn8 > tn3) && (tn8 > tn4) && (tn8 > tn5) && (tn8 > tn6)
&& (tn8 > tn7) && (tn8 > tn9) && (tn8 > tn10)
        plot(abs(fft(y7)));
        fprintf('2 VEN\n');
        TT8=1+TT8;
    end
    if (tn9 > tn1) && (tn9 > tn2) && (tn9 > tn3) && (tn9 > tn4) && (tn9 > tn5) && (tn9 > tn6)
&& (tn9 > tn7) && (tn9 > tn8) && (tn9 > tn10)
        plot(abs(fft(y8)));
        fprintf('2 YO\n');
        TT9=1+TT9;
    end
    if (tn10 > tn1) && (tn10 > tn2) && (tn10 > tn3) && (tn10 > tn4) && (tn10 > tn5) && (tn10 >
tn6) && (tn10 > tn7) && (tn10 > tn8) && (tn10 > tn9)
        plot(abs(fft(y9)));
        fprintf('2 ZAPATO\n');
        TT10=1+TT10;
    end
    %))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
    if (tm1 > tm2) && (tm1 > tm3) && (tm1 > tm4) && (tm1 > tm5) && (tm1 > tm6) && (tm1 >
tm7) && (tm1 > tm8) && (tm1 > tm9) && (tm1 > tm10)
        plot(abs(fft(y0)));
        fprintf('3 HAMBRE\n');
        TT1=1+TT1;
    end
```

```
    if (tm2 > tm1) && (tm2 > tm3) && (tm2 > tm4) && (tm2 > tm5) && (tm2 > tm6) && (tm2 >
tm7) && (tm2 > tm8) && (tm2 > tm9) && (tm2 > tm10)
        plot(abs(fft(y1)));
        fprintf('3 HIJO\n')
        TT2=1+TT2;
    end
    if (tm3 > tm1) && (tm3 > tm2) && (tm3 > tm4) && (tm3 > tm5) && (tm3 > tm6) && (tm3 >
tm7) && (tm3 > tm8) && (tm3 > tm9) && (tm3 > tm10)
        plot(abs(fft(y2)));
        fprintf('3 MARIA\n');
        TT3=1+TT3;
    end
    if (tm4 > tm1) && (tm4 > tm2) && (tm4 > tm3) && (tm4 > tm5) && (tm4 > tm6) && (tm4 >
tm7) && (tm4 > tm8) && (tm4 > tm9) && (tm4 > tm10)
        plot(abs(fft(y3)));
        fprintf('3 QUIERO\n');
        TT4=1+TT4;
    end
    if (tm5 > tm1) && (tm5 > tm2) && (tm5 > tm3) && (tm5 > tm4) && (tm5 > tm6) && (tm5 >
tm7) && (tm5 > tm8) && (tm5 > tm9) && (tm5 > tm10)
        plot(abs(fft(y4)));
        fprintf('3 ROPA\n');
        TT5=1+TT5;
    end
    if (tm6 > tm1) && (tm6 > tm2) && (tm6 > tm3) && (tm6 > tm4) && (tm6 > tm5) && (tm6 >
tm7) && (tm6 > tm8) && (tm6 > tm9) && (tm6 > tm10)
        plot(abs(fft(y5)));
        fprintf('3 SED\n');
        TT6=1+TT6;
    end
    if (tm7 > tm1) && (tm7 > tm2) && (tm7 > tm3) && (tm7 > tm4) && (tm7 > tm5) && (tm7 >
tm6) && (tm7 > tm8) && (tm7 > tm9) && (tm7 > tm10)
        plot(abs(fft(y6)));
        fprintf('3 TENGO\n');
        TT7=1+TT7;
    end
    if (tm8 > tm1) && (tm8 > tm2) && (tm8 > tm3) && (tm8 > tm4) && (tm8 > tm5) && (tm8 >
tm6) && (tm8 > tm7) && (tm8 > tm9) && (tm8 > tm10)
        plot(abs(fft(y7)));
        fprintf('3 VEN\n');
        TT8=1+TT8;
    end
    if (tm9 > tm1) && (tm9 > tm2) && (tm9 > tm3) && (tm9 > tm4) && (tm9 > tm5) && (tm9 >
tm6) && (tm9 > tm7) && (tm9 > tm8) && (tm9 > tm10)
        plot(abs(fft(y8)));
        fprintf('3 YO\n');
        TT9=1+TT9;
    end
    if (tm10 > tm1) && (tm10 > tm2) && (tm10 > tm3) && (tm10 > tm4) && (tm10 > tm5) &&
(tm10 > tm6) && (tm10 > tm7) && (tm10 > tm8) && (tm10 > tm9)
        plot(abs(fft(y9)));
```

```
        fprintf('3 ZAPATO\n');
        TT10=1+TT10;
    end
    %//////////////////////////////////////////////////////////////////////////////
    if (te1 > te2) && (te1 > te3) && (te1 > te4) && (te1 > te5) && (te1 > te6) && (te1 > te7)
&& (te1 > te8) && (te1 > te9) && (te1 > te10)
        plot(abs(fft(y0)));
        fprintf('4 HAMBRE\n');
        TT1=1+TT1;
    end
    if (te2 > te1) && (te2 > te3) && (te2 > te4) && (te2 > te5) && (te2 > te6) && (te2 > te7)
&& (te2 > te8) && (te2 > te9) && (te2 > te10)
        plot(abs(fft(y1)));
        fprintf('4 HIJO\n')
        TT2=1+TT2;
    end
    if (te3 > te1) && (te3 > te2) && (te3 > te4) && (te3 > te5) && (te3 > te6) && (te3 > te7)
&& (te3 > te8) && (te3 > te9) && (te3 > te10)
        plot(abs(fft(y2)));
        fprintf('4 MARIA\n');
        TT3=1+TT3;
    end
    if (te4 > te1) && (te4 > te2) && (te4 > te3) && (te4 > te5) && (te4 > te6) && (te4 > te7)
&& (te4 > te8) && (te4 > te9) && (te4 > te10)
        plot(abs(fft(y3)));
        fprintf('4 QUIERO\n');
        TT4=1+TT4;
    end
    if (te5 > te1) && (te5 > te2) && (te5 > te3) && (te5 > te4) && (te5 > te6) && (te5 > te7)
&& (te5 > te8) && (te5 > te9) && (te5 > te10)
        plot(abs(fft(y4)));
        fprintf('4 ROPA\n');
        TT5=1+TT5;
    end
    if (te6 > te1) && (te6 > te2) && (te6 > te3) && (te6 > te4) && (te6 > te5) && (te6 > te7)
&& (te6 > te8) && (te6 > te9) && (te6 > te10)
        plot(abs(fft(y5)));
        fprintf('4 SED\n');
        TT6=1+TT6;
    end
    if (te7 > te1) && (te7 > te2) && (te7 > te3) && (te7 > te4) && (te7 > te5) && (te7 > te6)
&& (te7 > te8) && (te7 > te9) && (te7 > te10)
        plot(abs(fft(y6)));
        fprintf('4 TENGO\n');
        TT7=1+TT7;
    end
    if (te8 > te1) && (te8 > te2) && (te8 > te3) && (te8 > te4) && (te8 > te5) && (te8 > te6)
&& (te8 > te7) && (te8 > te9) && (te8 > te10)
        plot(abs(fft(y7)));
        fprintf('4 VEN\n');
        TT8=1+TT8;
```

```
    end
    if (te9 > te1) && (te9 > te2) && (te9 > te3) && (te9 > te4) && (te9 > te5) && (te9 > te6)
&& (te9 > te7) && (te9 > te8) && (te9 > te10)
        plot(abs(fft(y8)));
        fprintf('4 YO\n');
        TT9=1+TT9;
    end
    if (te10 > te1) && (te10 > te2) && (te10 > te3) && (te10 > te4) && (te10 > te5) && (te10 >
te6) && (te10 > te7) && (te10 > te8) && (te10 > te9)
        plot(abs(fft(y9)));
        fprintf('4 ZAPATO\n');
        TT10=1+TT10;
    end

%=================================================================
===
    if TT1>=2
        msgbox('Identificado: HAMBRE','1','warn');
        [v fs]=wavread('hambre');
        soundsc(v,fs)
    end
    if TT2>=2
        msgbox('Identificado: HIJO','1','warn');
        [v fs]=wavread('hijo');
        soundsc(v,fs)
    end
    if TT3>=2
        msgbox('Identificado: MARIA','1','warn');
        [v fs]=wavread('maria');
        soundsc(v,fs)
    end
    if TT4>=2
        msgbox('Identificado: QUIERO','1','warn');
        [v fs]=wavread('quiero');
        soundsc(v,fs)
    end
    if TT5>=2
        msgbox('Identificado: ROPA','1','warn');
        [v fs]=wavread('ropa');
        soundsc(v,fs)
    end
    if TT6>=2
        msgbox('Identificado: SED','1','warn');
        [v fs]=wavread('sed');
        soundsc(v,fs)
    end
    if TT7>=2
        msgbox('Identificado: TENGO','1','warn');
        [v fs]=wavread('tengo');
        soundsc(v,fs)
    end
```

```
        if TT8>=2
           msgbox('Identificado: VEN','1','warn');
           [v fs]=wavread('ven');
           soundsc(v,fs)
        end
        if TT9>=2
           msgbox('Identificado: YO','1','warn');
           [v fs]=wavread('yo');
           soundsc(v,fs)
        end
        if TT10>=2
           msgbox('Identificado: ZAPATO','1','warn');
           [v fs]=wavread('zapato');
           soundsc(v,fs)
        end
        % ========================================
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
exit;
```