RESUMEN

El presente proyecto consiste en un control remoto por voz para el robot móvil Pioneer P3-DX que posee el Departamento de Eléctrica y Electrónica de la ESPE. Para realizar el control de la plataforma robótica se han integrado dos entornos de programación distintos, MATLAB escrito en lenguaje propietario y ARIA desarrollada en lenguaje C++.

La integración de estas dos aplicaciones es de trascendental importancia debido a las bondades que ambos poseen. Por un lado MATLAB que es un programa para análisis numérico de alta velocidad y procesamiento digital de señales, y por otro ARIA, que posee importantes librerías precompiladas para control de plataformas Pioneer. El programa implementado en MATLAB utiliza las herramientas de procesamiento de señales de manera que se elimine el ruido y se puedan extraer las características fundamentales de la voz para luego compararse con cada una de las características almacenadas en una base de datos. Para esta comparación se utiliza una medida de distancia Euclidiana. La señal almacenada con menor medida de distancia a la grabada representa la de mayor similitud.

CAPÍTULO 1

INTRODUCCIÓN

1.1 TEMA

CONTROL REMOTO POR VOZ DEL ROBOT MÓVIL PIONEER P3-DX.

1.2 DESCRIPCIÓN DEL PROYECTO

En el siglo XXI la tecnología se ha adelantado a la imaginación y ha convertido lo que eran sueños en algo posible, y de hecho el control por voz es en la actualidad un área en evolución que no se ha desarrollado por completo en el país. De allí la importancia en que este proyecto se convierta en el pionero en investigar y aplicar este tipo de control en la vida de las personas.

Los sistemas de reconocimiento de voz han tenido una gran demanda en la sociedad moderna, ante la necesidad de tener sistemas controlados de manera no física. Es por ello que reproducir las capacidades de los seres vivos ha sido una de las mayores metas para los investigadores de informática e inteligencia artificial. Como resultado de los años de estudio se han venido desarrollando técnicas muy útiles que han resuelto problemas específicos, aunque vale aclarar que estos no pueden llegar a competir con las capacidades de los seres vivos.

El proyecto a implementarse ayudará principalmente a seguir desarrollando nuevas alternativas de control de los robots móviles Pioneer. Hasta la actualidad se han venido realizando algunas prácticas con esta línea de robots, pero como alternativa novedosa hemos decidido realizar el control por voz para que el usuario sea capaz de conducir el robot mediante comandos sonoros.

La ejecución de acciones sin la necesidad de usar un joystick o cualquier otro dispositivo para control remoto hace que el ser humano se convierta en el controlador del robot, obligándolo a que éste obedezca lo que dictan sus palabras eliminando todo cableado común que exige al usuario estar junto al robot. Lo esencial es que dichas órdenes de voz podrán ahora viajar largas distancias, dependiendo del alcance del dispositivo de transmisión, que en nuestro caso particular será un micrófono inalámbrico tipo diadema.

Para poder comandar el robot se desarrolló una interfaz de comunicación entre MATLAB y la aplicación ARIA (Advanced Robotics Interface for Applications) que controla el microcontrolador.

Una vez lograda dicha integración, se aplica el control por voz al robot, ejecutando los comandos básicos de movimiento como: conectar, desconectar, adelante, atrás, paro, giro a la izquierda y giro a la derecha, todo esto puede ser manipulado y operado por el usuario mediante una interfaz desarrollada también en MATLAB.

El Sistema de Reconocimiento de Voz permite que el usuario grabe una palabra por medio de un micrófono y ésta sea reconocida en la base de datos. El sistema posee un entorno gráfico, que proporciona la opción de mando por voz, donde la señal de voz es ingresada y procesada por los algoritmos del programa, obteniendo los parámetros más significativos de la señal de voz, para luego ser almacenados en la computadora. El proceso de reconocimiento permite que la palabra sea comparada con la base de datos almacenada.

El entorno gráfico permite por otra parte visualizar las palabras grabadas y reconocidas. Debido que el Sistema de Reconocimiento de Voz procesa digitalmente las señales, el análisis gráfico en el reconocimiento puede hacerse mediante un análisis del espectro de frecuencias de la señal de voz.

El presente trabajo es una recopilación de los métodos de procesamiento digital y una explicación de los algoritmos utilizados en el programa del Sistema de Reconocimiento de Voz. Son explicados varios métodos de procesamiento digital de voz y los recursos necesarios para la elaboración del sistema. Además, se incluye la explicación del funcionamiento de la interfaz MATLAB-ARIA que permite al sistema de reconocimiento de voz comandar al robot Pioneer P3-DX, terminando con un esquema gráfico general del programa, la evaluación del mismo.

CAPÍTULO 2

MARCO TEÓRICO

2.1 PLATAFORMAS PIONEER

Pioneer es una familia de robots móviles de dos o cuatro ruedas, entre los cuales podemos encontrar al Pioneer 1, Pioneer AT, Pioneer 2-DX, -DXe, -DXf, -CE, -AT, el Pioneer 2-DX8/Dx8 Plus, AT8/AT8 Plus, y los nuevos robots Pioneer 3-DX y 3-AT.

Las plataformas Pioneer DX integran a robots para la educación, investigación, creación de prototipos, exhibiciones y otros proyectos. Los robots Pioneer son pre-ensamblados, personalizados, modernizados y fabricados lo suficientemente resistentes para durar por años en laboratorios y en aulas de estudio. Su versatilidad, confiabilidad y durabilidad los ha hecho la plataforma preferida dentro de la familia de robots inteligentes.

Estos pequeños robots de desarrollo e investigación comparten una arquitectura y software común para todas las plataformas Mobile-Robots, lo cual incluye al AmigoBot™, PeopleBot™ V1, PeopleBot™ y PowerBot™.

Las plataformas MobileRobots son un estàndar en robots móviles, puesto que contienen todos los componentes básicos para realizar el sensado y la navegación en ambientes reales. Estos robots han sido intérpretes en algunos proyectos importantes de investigación a nivel mundial, entre ellos encontramos a la Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA) de los Estados Unidos..

La plataforma MobileRobot también trae un Host para hacer aplicaciones con un software cliente que permite el control de la plataforma y el desarrollo de aplicaciones según el entorno. El desarrollo del software incluye su interfaz propia: ARIA (Advanced Robotics Interface for Aplications), así como ARNetworking, ambas publicadas bajo licencia pública GNU/Linux, con librerías completamente documentadas en C, Java y Python; así como el código fuente de las mismas. (Morales & Daniel, 2010)

2.1.1 Pioneer 3-DX[1]

El Pioneer 3-DX es un pequeño y ligero robot de dos ruedas, y dos motores ideales para el uso dentro de un laboratorio o en un aula de clases. La figura 2.1 muestra el robot Pioneer P3-DX.

Figura. 2.1 Robot Pioneer 3-DX

[1] Pioneer 3 Operations Manual

La plataforma robótica P3-DX cuenta con muchas características interesantes. Funciona bajo una arquitectura denominada cliente – servidor, el cual es un modelo de comunicación en el que el microcontrolador del robot interpreta las órdenes de un programa (cliente) localizado en una PC aparte, el cual puede ser una laptop o una PC de escritorio o un PC interno. Algunas especificaciones técnicas mostradas en el manual de operaciones del robot Pioneer P3-DX (MobileRobots, 2006) se encuentran descritas en la tabla 2.1.

ESPECIFICACIONES	
Construcción	Cuerpo: 1.6 mm aluminio
	Neumáticos: Rellenas de espuma de goma
Operación	Peso: 9kg
	Carga útil: 17kg
Movimientos	Radio de giro: 0cm
	Radio de rotación: 26.7cm
	Vel. máxima hacia adelante/ hacia atrás: 1.2 m/s
	Vel. de rotación: 300º/s
	Max. Paso: 2,5 cm
	Max. Espacio de desplazamiento: 5 cm
	Max. Grado de desplazamiento: 25%
	Terreno: cubierta, interiores
Energía	Tiempo de ejecución: 8-10 horas / 3 baterías (sin accesorios)
	Tiempo de carga: 12 horas (estándar) o 2,4 horas (opcional de alta capacidad de carga)
	Suministros disponibles de energía: 5 V a 1.5 Fuente conmutada 12 V a 2.5 Fuente conmutada
Baterías	Soporta hasta 3 baterías a la vez
	Tensión: 12 V
	Capacidad: 7,2 A (cada una)
	Baterías reemplazables en caliente: Sí

Opciones de recarga	Punto directo de conexión
	Estación de acoplamiento Powercube (3-puertos de carga de batería) * Las baterías son accesibles a través de bisagras aseguradas al panel para conexión en caliente (funcionamiento continuo)
E/S del Microcontrolador	32 entradas digitales
	8 salidas digitales 7 entradas analógicas
	3 puertos serie de expansión (algunos puertos pueden no incluirse si se encuentran disponibles otros accesorios del robot)
Panel de control del usuario	Indicador principal de poder
	Botón de habilitación de motores
	Indicador de carga de batería
	2 switches AUX
	Reseteo del sistema

Tabla. 2.1. Especificaciones técnicas del robot Pioneer 3-DX

2.1.2 ARIA (Interfaz Robótica Avanzada para Aplicaciones)

ARIA (Interfaz Robótica Avanzada para Aplicaciones) es una interfaz orientada a objetos, es una Interfaz de programación de aplicaciones (API) para MobileRobots de la línea de robots inteligentes, incluyendo el Pioneer 2.3 DX y AT, PeopleBot, PowerBot, AmigoBot, PatrolBot / Guiabot, Seekur y SeekurJr.

ARIA está escrita en el lenguaje C y trata de ayudar al cliente de la plataforma a un rápido acceso y gestión del servidor de robot, así como a los sensores y muchos otros accesorios de gran utilidad.

ARIA se puede ejecutar múltiples o simples rutinas a la vez, bajo las plataformas Linux y Win32. Puede usarse para ejecutar acciones diferentes, desde simples comandos

de control del servidor para un accionamiento directo del robot, hasta un alto nivel de programación para realizar acciones inteligentes. Además se puede agregar sus propias características y modificaciones a ARIA debido a que es de código abierto.

El paquete ARIA incluye tanto el código fuente y las bibliotecas pre-compiladas, como algunos programas de ejemplo. Estas bibliotecas y programas se construyen con el compilador GCC 3.4 si en Linux, y Microsoft Visual Studio. NET 2003 (7.1), Visual C Express 2008 (9.0) o Visual C Express 2010 (10,0) si es Windows. Si utiliza un compilador diferente o una versión diferente del compilador, se debe reconstruir las bibliotecas de ARIA para asegurar la compatibilidad del enlace.

ARIA soporta robots de diferentes formas y tamaños. El robot más pequeño soportado es el AmigoBot, el cual pesa 3.6 kilogramos y está equipado con ocho sonares. Por otro lado encontramos el PowerBot, que es el robot más grande soportado, el cual pesa 120 kilogramos y está equipado con 28 sonares, mapeo por láser y un brazo de sujeción o gripper. La mayoría, si no todos los robots fabricados por ActiveMedia son compatibles con la librería ARIA. La figura 2.2 muestra los robots soportados por ARIA. (Borgstrom, 2005)

ROBOT	APLICACIÓN
AmigoBotTM	En aulas y equipos de robots.
Pioneer 3-AT	Robot de alto rendimiento en todo terreno.
Pioneer 3-DX	Robot educativo y de investigación.
PatrolBotTM	Robot de vigilancia.
PowerBotTM	Alta agilidad, robot de alta capacidad de carga.
PeopleBotTM	Robot de interfaz humana.

Tabla. 2.2 Robots soportados por ARIA

2.1.3 ARIA-API

ARIA cuenta con un API (Interfaz de programación de aplicaciones) grande y poderosa. El API contiene un conjunto de funciones de alto nivel que permite a los desarrolladores controlar el robot usando tanto funciones incorporadas como funciones propias definidas por el usuario. Permite a los programadores definir nuevas acciones por medio de subclases ArAction. Estas acciones se ejecutan en un subproceso de segundo plano permitiendo que la aplicación principal haga frente a otros problemas que se relacionan con el comportamiento básico del robot. También es posible controlar directamente los movimientos y sensores del robot usando un conjunto de funciones de bajo nivel. Estas funciones comandan el robot para realizar diferentes tareas como "Dar la lectura de la distancia medida por el sensor X", "Moverse hacia adelante con una velocidad Y ", etc. ARIA también contiene un API de aún más bajo nivel que permite a los usuarios comunicarse y enviar comandos directamente a la computadora a bordo del robot. La figura 2.2 muestra el API de ARIA.

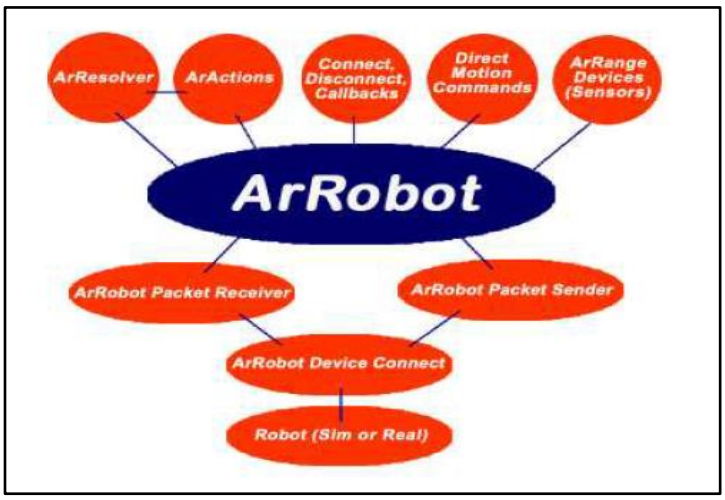


Figura. 2.2 Diagrama de bloques del API de ARIA

2.1.4 Software de simulación MOBILESIM

MobileSim es un software para la simulación de plataformas MobileRobots/ActivMedia y sus ambientes de exploración, muy útil para corregir errores de programación cuando se trabaja con ARIA. Éste sustituye al SRIsim distribuido previamente con ARIA.

MobileSim se basa en el simulador Stage, creado por Richard Vaughan, Andrew Howard, y otros, como parte del proyecto Player/Stage, con algunas modificaciones hechas por MobileRobots.

MobileSim usa datos de línea de un mapa (.map) para simular paredes y otros obstáculos en el entorno. Existen otras aplicaciones que nos permiten crear estos mapas, como por ejemplo Mapper3 o Mapper3-Basic, pero no son difundidos en forma gratuita, es por ello que estos mapas también pueden crearse usando un simple editor de texto como Notepad o Wordpad.

MobileSim ha sido probado en Windows 2000, Windows XP, RedHat GNU/Linux 7.3, Debian GNU/Linux 3.1 y Debian GNU/Linux 5.0. Además puede descargarse gratuitamente de la dirección electrónica: http://robots.mobilerobots.com/MobileSim/.

2.2 MATLAB

MATLAB es el nombre abreviado de “MATrix LABoratory”. MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares tanto reales como complejos, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las

capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio.

El lenguaje está construido por código llamado M-code que puede ser fácilmente ejecutado en la ventana de comandos. Con lo cual se pueden crear funciones, etc. Pero la razón principal para la elección de este lenguaje de programación son las herramientas que proporciona para el procesamiento de señales, y el conjunto de funciones para el procesamiento digital

El lenguaje de programación de MATLAB siempre es una magnífica herramienta de alto nivel para desarrollar aplicaciones técnicas, fácil de utilizar y que aumenta significativamente la productividad de los programadores respecto a otros entornos de desarrollo.

Además, para crear entornos gráficos se puede utilizar el GUIDE de MATLAB, que provee herramientas para crear GUIs (Graphical User Interface), con lo cual se puede crear la forma del entorno grafico así como asociar funciones a los elementos del GUI. MATLAB también incluye funciones para manipular archivos.

2.2.1 Funciones MEX[2]

Es posible compilar código C, C++, o Fortran haciendo llamadas desde MATLAB. Este tipo de programación es llamada MATLAB Ejecutable (MEX) o más comúnmente conocida como funciones MEX. Estas funciones MEX permiten un alto nivel de programación en C, C++, y Fortran mientras se trabaja dentro del ambiente de MATLAB (Getreuer, 2010). La figura 2.3 ilustra la puerta de enlace entre MATLAB con una aplicación escrita en lenguaje C++.

[2] Writing MATLAB C/MEX Code

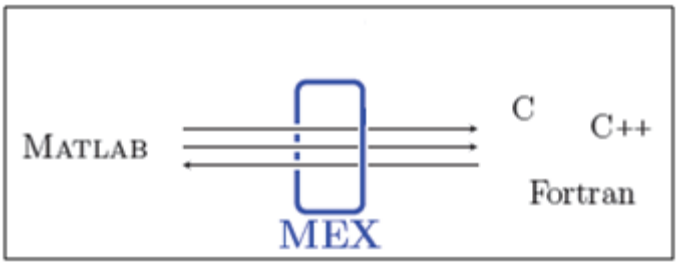


Figura. 2.3. Conversión MATLAB-C++

MATLAB puede ser integrado con otros lenguajes y aplicaciones de dos formas diferentes:

- Mediante una aplicación escrita en C o Fortran que puede utilizar MATLAB como herramienta de cálculo mediante el uso de una librería. Esta librería le permite a la aplicación comunicarse con MATLAB mediante una interfaz-COM en caso de las plataformas Windows.

- La otra forma es mediante funciones en lenguajes como C y Fortran, las cuales pueden ser escritas y usadas desde el propio MATLAB. Esto se consigue mediante el uso de Archivos-MEX. Un Archivo-MEX consta de dos partes distintas. La primera consiste de una rutina de programación donde esté implementada la función requerida, la cual puede contener un algoritmo matemático complejo, una rutina de comunicación con algún hardware sofisticado o algo completamente diferente. La otra parte es la rutina de enlace. Su responsabilidad es ser la interfaz entre la parte de programación y MATLAB.

La razón principal por la cual las personas algunas veces usan extensiones MEX dentro de un código de MATLAB es por el rendimiento que se logra tener. Implementar difíciles algoritmos computacionales en lenguajes como C/C++ o Fortran puede conducir a mejoras significativas de rendimiento.

También se requieren extensiones MEX cuando se accede a algunas funciones que dependen del sistema operativo y el uso de bibliotecas externas, o al comunicarse directamente con el hardware.

Si no se aplica correctamente las bondades de una función MEX pueden convertirse en un problema antes que en una solución. Es por ello que se recomienda que antes de usar una función MEX se optimice completamente el código en MATLAB y luego se use de preferencia una función MEX para sustituir ciertos problemas o cuellos de botella que se puede tener al programar en MATLAB. Es evidente que si todas las funciones se convierten en MEX, es preferible escribir la aplicación entera en código C/C++.

Para tener una idea más clara de cómo funciona y se crea una función MEX, explicaremos un breve ejemplo:

– Colocar el código que se muestra en la figura 2.4 dentro del editor de MATLAB o algún editor de código C y guardarlo como hello.c.

hello.c

```
#include "mex.h"  /* Always include this */

void mexFunction(int nlhs, mxArray *plhs[],      /* Output variables */
                 int nrhs, const mxArray *prhs[]) /* Input variables  */
{
    mexPrintf("Hello, world!\n");  /* Do something interesting */
    return;
}
```

Figura. 2.4. Ejemplo de función MEX

– Compilamos el código desde MATLAB ingresando el siguiente comando
>> mex hello.c

Si todo está correcto, habremos programado un bloque de código en lenguaje C que ahora puede ser usado dentro de la plataforma de MATLAB. El archivo que se genera luego de esta compilación se llama hello.mexw32 (o similar, dependiendo de la plataforma).

El compilar en C requiere que tenga instalado y correctamente configurado algún compilador adecuado dentro de MATLAB para poder usarlo. MATLAB detecta los compiladores más populares, incluyendo el Microsoft Visual C/C++ y el GCC.

Algunas versiones de MATLAB vienen con el compilador de código C llamado Lcc, pero recordemos que la aplicación a desarrollar debe estar programada en código C++ para ser compatible con el software ARIA.

- Si llamamos desde MATLAB a la función MEX, tendremos como resultado lo siguiente:

```
>> hello

Hello, world!
```

Debe considerarse que la función MEX compilada puede no ser compatible entre diferentes plataformas o diferentes versiones de MATLAB. Éstas deberían ser compiladas para cada versión en la que se use. Es importante aclarar que MATLAB viene con ejemplos de funciones MEX dentro del directorio MATLAB/extern/examples.

Para comprender las entradas y salidas que posee una función MEX se puede analizar la siguiente línea de código:

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

Aquí “mxArray” es un tipo de representación de una variable de MATLAB. La tabla 2.3 muestra los argumentos de una función MEX.

C/MEX	SIGNIFICADO	CÓDIGO .M
Nlhs	Número de variables de salida	Nargout
Plhs	Arreglo de punteros mxArray para las variables de salida	Varargout
Nrhs	Número de variables de entrada	Nargin
prhs	Arreglo de punteros mxArray para las variables de entrada.	varargin

Tabla. 2.3 Argumentos de una función MEX en código C++ y en lenguaje M

Las variables MEX son análogas a las variables en código .M como nargout, varargout, nargin, y varargin. El nombre “\lhs" es una abreviación de left-hand side (variables de salida) y “\rhs" es una abreviación de right-hand side (variables de entrada). Por ejemplo, suponiendo que la función MEX es llamada de la siguiente manera:

[X,Y] = mymexfun(A,B,C)

Entonces nlhs = 2 (número de variables de salida) y plhs [0] y plhs[1] son arreglos de punteros para las variables de salida (tipo mxArray) apuntando respectivamente a X y Y. Similarmente las entradas estarían dadas por rlhs = 3 con prhs[0], prhs[1], y prhs[2] como arreglos de punteros de variables de entrada para A, B, y C respectivamente.

Las variables de salida están inicialmente no asignadas, es responsabilidad de la función MEX crearlas. Si nlhs = 0 (número de variables de salida), la función MEX retorna por sí sola una variable de salida, en cuyo caso plhs[0] representa a esta variable. (Getreuer, 2010)

2.3 SISTEMA DE RECONOCIMIENTO DE VOZ

2.3.1 Señal de voz

2.3.1.1 Breve anatomía del aparato fonador[3]

La voz humana se produce por medio del aparato fonador. Este está formado por los pulmones, que representan la fuente de energía, la laringe, que contiene las cuerdas vocales, la faringe, las cavidades oral y nasal, y una serie de elementos articulatorios: los labios, los dientes, el alveolo, el paladar, el velo del paladar y la lengua.

La figura 2.5 muestra la composición anatómica del aparato fonador.

[3] El Aparato Fonador

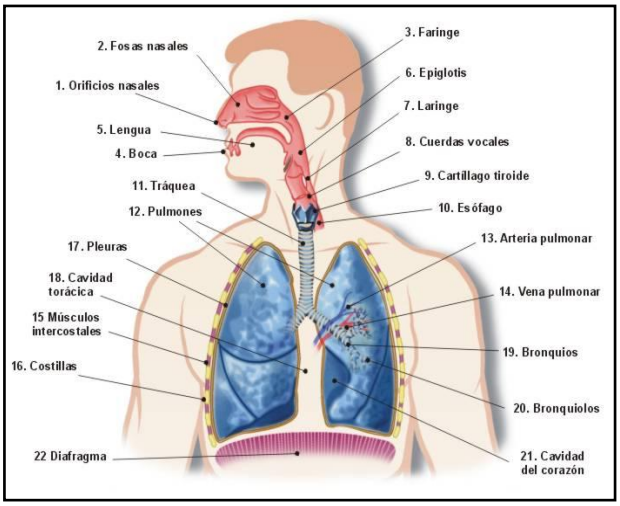


Figura. 2.5. Anatomía del aparato fonador

La voz se produce por la vibración de las cuerdas vocales cuando se acercan entre sí como consecuencia del paso del aire a través de la laringe.

La laringe es el órgano más importante de la voz, y se encuentra localizada en la parte central del cuello y en el tramo final de la tráquea. Está formada básicamente por músculos y cartílagos. En su interior se encuentran las cuerdas vocales (también llamadas pliegues vocales), porque en realidad, no tienen forma de cuerda, sino que se trata de una serie de repliegues o labios membranosos, son dos bandas de tejido muscular que se insertan en los cartílagos.

Al abrirse se respira y al cerrarse se produce la fonación. Las cuerdas vocales pueden tensarse o distenderse, lo que producirá sonidos agudos en el primer caso, y graves en el segundo.

Si se abren y se recogen a los lados, el aire pasa libremente, sin hacer presión: respiramos. Si, por el contrario, se juntan, el aire choca contra ellas, que vibran a modo de lengüetas, produciendo un sonido tonal. La frecuencia de este sonido depende del tamaño y tensión de las cuerdas, y de la velocidad del flujo del aire proveniente de los pulmones.

Los movimientos de los cartílagos de la laringe permiten variar el grado de apertura entre las cuerdas y una depresión o una elevación de la estructura laríngea, con lo que varía el tono de los sonidos producidos por el paso del aire a través de ellos. Esto junto a la disposición de los otros elementos de la cavidad oral (labios, lengua y boca) permite determinar los diferentes sonidos que emitimos.

El sonido que producen las cuerdas vocales es muy débil, resultaría insignificante e incluso desagradable, a no ser por la caja de resonancia que lo amplifica, y le otorga el timbre, adquiriendo forma. (Gutiérrez, 2012)

La figura 2.6 muestra el proceso de producción de la voz.

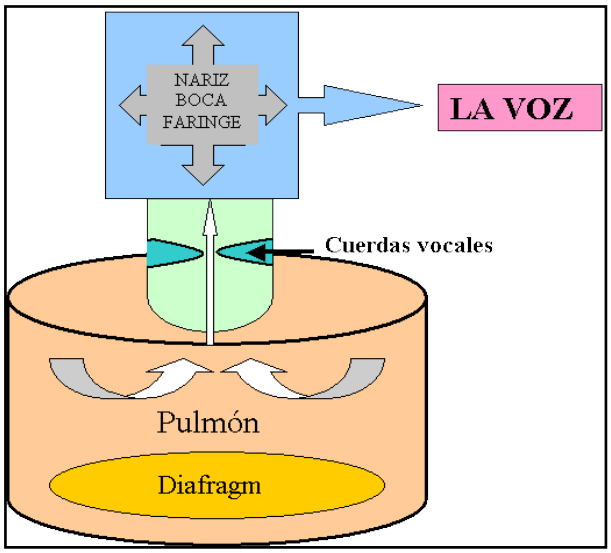


Figura. 2.6 Proceso de producción de la voz

2.3.1.2 Modelo del tracto vocal

El tracto vocal modelado se manifiesta como un filtro variable en el tiempo cuyos parámetros varían en el tiempo en función de la acción consciente que se realiza al pronunciar una palabra.

El filtro variable en el tiempo tiene dos posibles señales de entrada que dependerán del tipo de señal, sonora o sorda (no sonora). Para señales sonoras la excitación será un tren de impulsos de frecuencia controlada, mientras que para las señales no sonoras la excitación será ruido aleatorio.

La combinación de estas señales modela el funcionamiento de la glotis. El espectro de frecuencias de la señal vocal puede obtenerse a partir del producto del espectro de la excitación por la repuesta en frecuencia del filtro.

El tracto vocal manifiesta un número muy grande de resonancias, sin embargo se consideran solo las tres o cuatro primeras que toman el nombre de 'formantes' y cubren un rango de frecuencias entre 100 y 3500 Hz. Esto es debido a que las resonancias de alta frecuencia son atenuadas por la característica frecuencial del tracto que tiende a actuar como un filtro pasabajo con una caída de aproximadamente -6 dB por octava (Gómez, 2011).

La figura 2.7 muestra el diagrama de bloques del funcionamiento del tracto vocal en el proceso de generación de la voz.

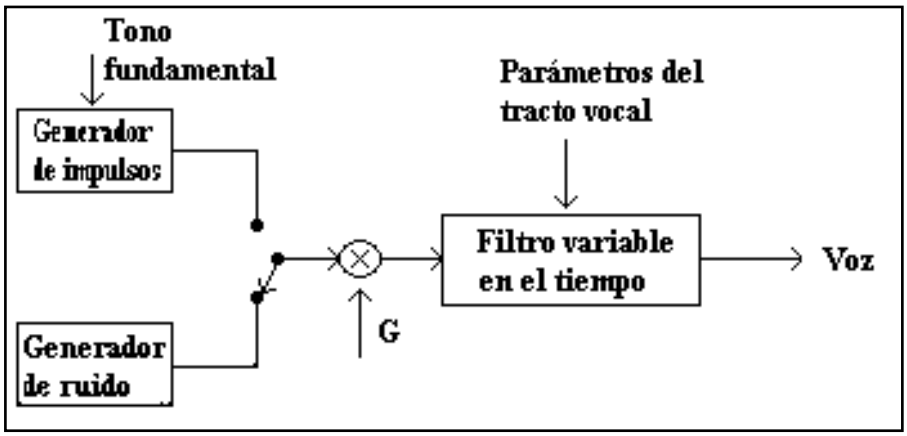


Figura. 2.7 Modelo del tracto vocal

Este modelo es una simplificación del proceso del habla. Los sonidos fricativos no se filtran por el tracto con la misma extensión en que lo hacen las señales sonoras por lo que el modelo no es muy preciso para este tipo de señales. Además el modelo supone que las dos señales pueden separarse sin considerar ninguna interacción entre ellas, lo cual no es cierto ya que la vibración de cuerdas vocales es afectada por las ondas de presión dentro del tracto. Sin embargo estas consideraciones pueden ser ignoradas resultando el modelo lo suficientemente adecuado.

2.3.1.3 Obtención de información mediante micrófono

El audio es un fenómeno analógico. Para grabar una señal de voz se hace la conversión de la señal analógica del micrófono en una señal digital por medio del conversor A/D en la tarjeta de sonido. Cuando un micrófono está operando las ondas de sonido hacen que vibre el elemento magnético del micrófono causando una corriente eléctrica hacia la tarjeta de sonido, donde el conversor A/D básicamente graba los voltajes eléctricos en intervalos específicos. Hay dos factores importantes durante este proceso. Primero está la tasa de muestreo o que tan seguido los valores de voltaje son grabados. Segundo, son los bits por segundo, o que tan exactamente los valores son grabados. Un tercero podría ser el número de canales (mono o estéreo), pero para las aplicaciones de reconocimiento de voz un canal mono es suficiente.

2.3.1.4 Micrófono empleado en el sistema de reconocimiento de voz

El sistema de reconocimiento de voz diseñado, emplea el micrófono inalámbrico Wireless tipo diadema marca EB-330H para la recepción de la voz mostrado en la figura 2.8.

Figura. 2.8 Micrófono usado en la aplicación

Las especificaciones técnicas de este dispositivo se muestran en la tabla 2.4.

MICRÓFONO PROFESIONAL CON SISTEMA WIRELESS EB-330H	
CARACTERÍSTICAS DE FUNCIONAMIENTO	
Total de canales	1 canal
Rango dinámico	Más de 110dB
Distorsión armónica total	Menos de 0.5%
Respuesta de frecuencia	80Hz-18kHz ($\pm 3dB$)
Relación señal/ruido	Más de 90dB

Rango efectivo	Más de 80m
TRANSMISOR	
Requerimientos de potencia	<10mW
Tipo de Modulación	FM
Imagen y rechazo de falsos	Más de 50dB
Frecuencia de trabajo	231.8 MHz
Alimentación	9VDC
RECEPTOR	
Sensibilidad	-107dBm
Tipo de transductor mecano-acústico	Oscilador de cristal
Relación señal/ruido	Más de 90dB
Distorsión armónica total	Menos de 0.5%
Voltaje de salida	500mv
Frecuencia de trabajo	231.8 MHz
Alimentación	3-4.5 VDC

Tabla. 2.4 Datos técnicos del micrófono EB-330H

2.3.2 Procesamiento digital de señales[4]

El procesamiento digital de señales o DSP (digital signal processing) es la manipulación matemática de una señal de información para modificarla o mejorarla en algún sentido. Éste está caracterizado por la representación en el dominio del tiempo discreto, en el dominio de la frecuencia discreta, u otro dominio discreto de señales por medio de una secuencia de números o símbolos y el procesado de esas señales.

Se puede procesar una señal para obtener una disminución del nivel de ruido, para mejorar la presencia de determinados matices, como los graves o los agudos y se realiza

[4] Sistema de reconocimiento de voz en MATLAB

combinando los valores de la señal para generar otros nuevos. Así, el DSP se utiliza en el procesamiento de música (por ejemplo MP3), de voz (por ejemplo, reconocimiento de voz), de imágenes (en la transmisión de imágenes satelitales) y vídeo (DVDs).

2.3.2.1 Transformada rápida de Fourier

La transformada rápida de Fourier tiene mucha importancia en una gran variedad de aplicaciones que requieren un procesamiento digital de señales. FFT es la abreviatura usual (de sus siglas en inglés Fast Fourier Transform), y es un eficiente algoritmo que permite calcular la transformada discreta de Fourier y su inversa dados vectores de longitud N por las ecuaciones 2.1 y 2.2 respectivamente.

$$x_k = \sum_{n=0}^{N-1} x_n e^{-jk\frac{2\pi}{N}n}, \quad k = 0, \dots, N-1. \qquad \text{Ecuación 2. 1}$$

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} x_k e^{jk\frac{2\pi}{N}n}, \quad n = 0, \dots, N-1. \qquad \text{Ecuación 2. 2}$$

2.3.2.2 Muestreo y Teorema de Nyquist

MUESTREO

El muestreo consiste en el proceso de conversión de señales continuas a señales discretas en el tiempo, es un paso para digitalizar una señal analógica. Este proceso se realiza midiendo la señal en momentos periódicos del tiempo con el Teorema de Nyquist.

TEOREMA DE NYQUIST

Para interpretar este teorema debemos considerar a x[n] como una secuencia de muestras obtenida a partir de una señal continua en el tiempo x(t) por medio de la relación mostrada en la ecuación 2.2

$$x[n] = x(nT), \ \ para \ \ -\infty \leq n \leq \infty \qquad \text{Ecuación 2.3}$$

Donde T es el período de muestreo, y su reciproco, $f_s = \frac{1}{T}$ es la frecuencia de muestreo, en muestras por segundo. También podemos expresar la frecuencia de muestreo como $\Omega_s = \frac{2\pi}{T}$ en radianes por segundo.

Entonces el teorema de muestreo de Nyquist está definido como: sea x(t) una señal limitada en banda dada por la ecuación 2.4

$$X(j\Omega) = 0 \ \ \text{para } |\Omega| \geq \Omega_{\mathrm{N}} \qquad \text{Ecuación 2.4}$$

Entonces x(t) está únicamente determinada por sus muestras $x[n] = x(nT)$, $n = 0, \pm 1, \pm 2, \ldots$ si $\Omega_s = \frac{2\pi}{T} \geq 2\Omega_N$.

La frecuencia Ω_N es comúnmente referida como la frecuencia de Nyquist, y la frecuencia $2\Omega_N$ que tiene que ser excedida por la frecuencia de muestreo es llamada la razón de Nyquist tal como se muestra en la figura 2.9.

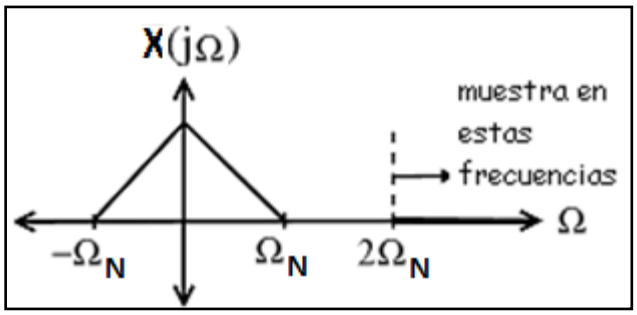


Figura. 2.9 Ilustración de la frecuencia de Nyquist

2.3.2.3 Eliminación de ruido

Durante esta etapa se escanea la señal digitalizada y se remueven las zonas de silencio por medio del cálculo de la energía en corto tiempo. Para este propósito se escogieron segmentos de 10ms. Si en un segmento la energía promedio es menor que un valor umbral proporcional a la energía promedio de la señal entera, dicho segmento es descartado. El valor umbral escogido es 20%. (Velásquez, 2008)

Se empleó la ecuación 2.5 para el cálculo de la energía promedio de la señal:

$$E_{avg} = \frac{1}{N}\sum_{k=1}^{N}|x[k]|^2 \qquad \text{Ecuación 2.5}$$

2.3.2.4 Filtro de Pre-énfasis

Para continuar con el análisis se hace necesario realizar un preprocesamiento de la señal vocal mediante la técnica de preénfasis

La etapa de preénfasis 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 (Velásquez, 2008). Se usa un filtro digital de primer orden cuya función de transferencia está definida por la ecuación 2.6.

$$H(z) = \frac{Y(z)}{X(z)} = (1 - az^{-1}) \quad , a = 0.95 \; en \; nuestro \; caso \qquad \text{Ecuación 2.6}$$

Y la fórmula en diferencias correspondiente a la función de transferencia es la mostrada en la ecuación 2.7.

$$y[n] = x[n] - ax[n-1] \qquad \text{Ecuación 2.7}$$

La representación de esta ecuación en un diagrama de bloques se muestra en la figura 2. 10.

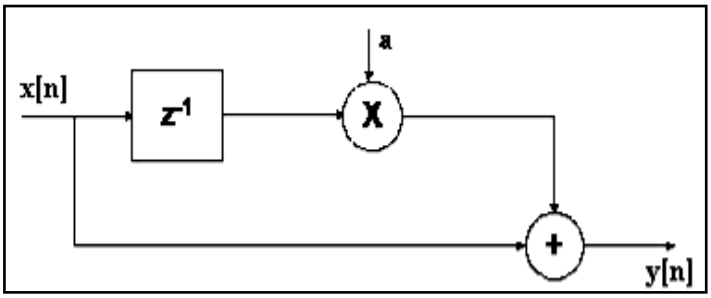


Figura. 2.10 Diagrama de bloques del filtro de pre-énfasis

La gráfica de la figura 2.11 muestra la forma del filtro de preénfasis en el dominio frecuencial.

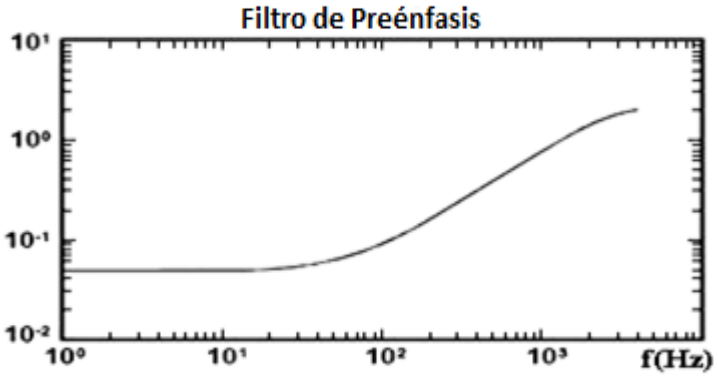


Figura. 2.11 Respuesta frecuencial del filtro de pre-énfasis

2.3.2.5 Segmentación y ventaneo

SEGMENTACIÒN[5]

Debido a la inercia inherente a los órganos articulatorios se puede suponer que las características y propiedades de la señal de voz no varían apreciablemente en un intervalo corto de tiempo, que aproximadamente oscila entre [20 30ms], por lo cual es posible realizar un análisis espectral cuasi-estacionario sobre segmentos de señal de esta duración temporal.

La evolución temporal de las características espectrales se obtiene repitiendo el análisis sobre estos segmentos consecutivos de la señal, que suelen tomarse con un cierto solapamiento temporal que va desde el 50% al 70% del segmento previo. El resultado del proceso de cada ventana es uno o varios parámetros que producen una sucesión nueva, que evoluciona en el tiempo. Esta parametrización de cada ventana se utiliza como una representación de la señal de voz. La figura 2.12 muestra el enventanado de una señal sinusoidal. (García, 2010)

[5] Sistema de Reconocimiento de Palabras para un robot instrumentista

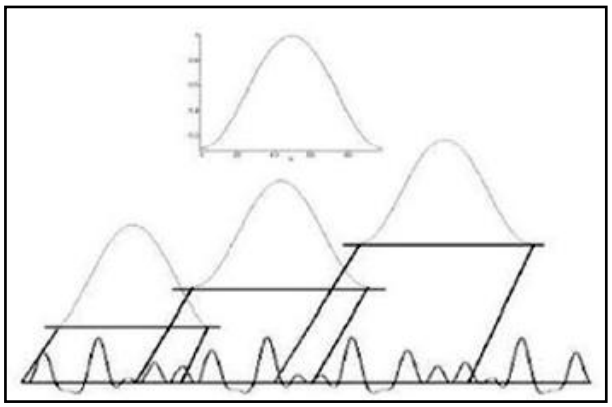


Figura. 2.12 Enventanado de un segmento de señal sinusoidal

Para complementar el proceso de segmentación es recomendable aplicar una técnica de ventaneo apropiada. Hay muchos tipos de ventanas: rectangular, Hamming, Hanning, Blackman, Bartlett, y Kaiser. Hoy en día, en reconocimiento de voz, se utiliza exclusivamente la ventana Hamming, que es un caso específico de la Hanning.

VENTANEO

En el proceso de procesamiento de voz, se asume que la señal es estacionaria en intervalos de tiempo lo suficientemente cortos. Durante la pronunciación de un fonema la señal es cuasiestacionaria, nosotros asumiremos que es estacionaria en tramos pequeños, lo cual es útil para el análisis de estos “trozos de voz estacionarios”, ya que se realizan Transformadas de Fourier a cada intervalo. La solución para obtener los intervalos o trozos es por medio del ventaneo, que consiste en multiplicar la señal por una función ventana cuyo valor fuera de un determinado rango es cero Es importante analizar el efecto de cada una de las ventanas, ya que permite disminuir los efectos de las discontinuidades, los tipos de ventanas más conocidas son:

1. Ventana Rectangular

$$w_n = \begin{cases} 1 & 0 \leq n \leq N \\ 0 & otro\ valor \end{cases}$$ Ecuación 2.8

La forma de una ventana rectangular se observa en la figura 2. 13

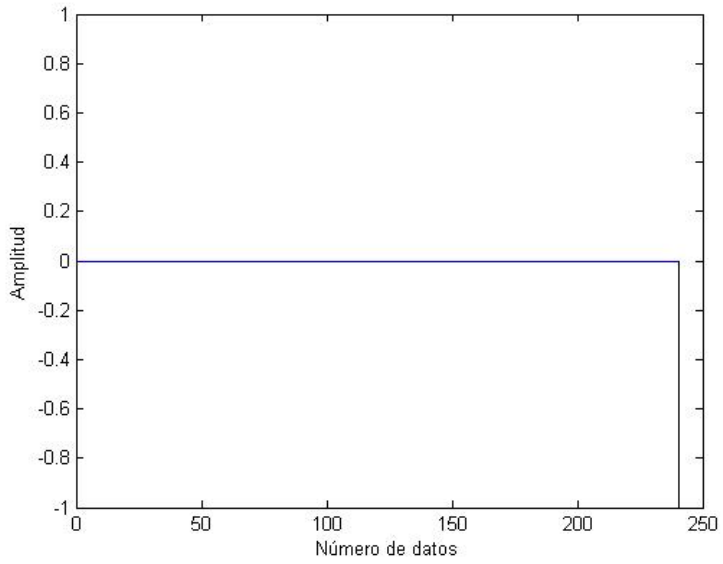


Figura. 2.13 Representación gráfica de la ventana rectangular

Pero hay que considerar las discontinuidades que esto produce en los extremos de la señal, las distorsiones, por otro lado el parámetro de la amplitud se ve siempre un poco afectado.

2. Ventana Hanning

$$vh = 0.5\left(1 - cos\left(2\pi\frac{k}{n+1}\right)\right), k = 1, \ldots n$$ Ecuación 2.9

La forma de una ventana de Hanning se observa en la figura 2. 14

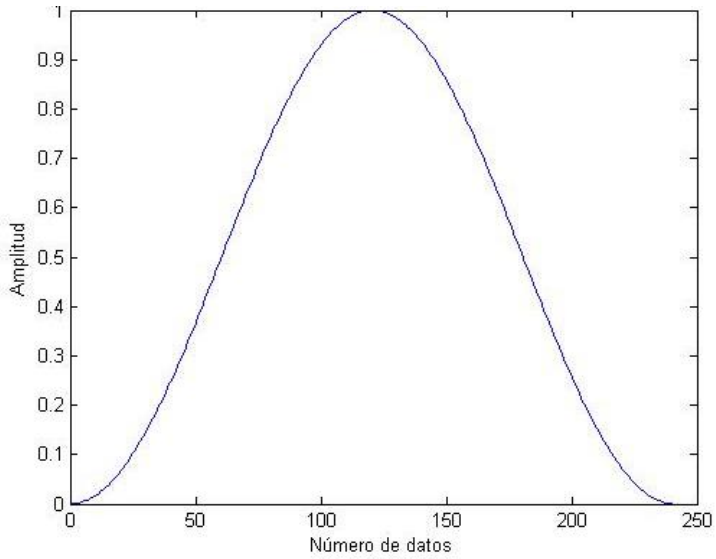


Figura. 2.14 Representación gráfica de la ventana Hanning

Esta ventana tiene un efecto en el dominio del tiempo y de la frecuencia. En el dominio del tiempo podemos decir que la ventana hace disminuir la amplitud de la señal cerca de los bordes de la ventana lo cual ayuda a eliminar las discontinuidades.

3. Ventana Hamming: esta es la ventana más común en el análisis de señales de voz, se define por la ecuación 2. 10.

$$w_n = \begin{cases} 0.54 - 0.46cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N \\ 0 & otro\ valor \end{cases} \qquad \text{Ecuación 2. 10}$$

La forma de una ventana de Hamming se observa en la figura 2.15

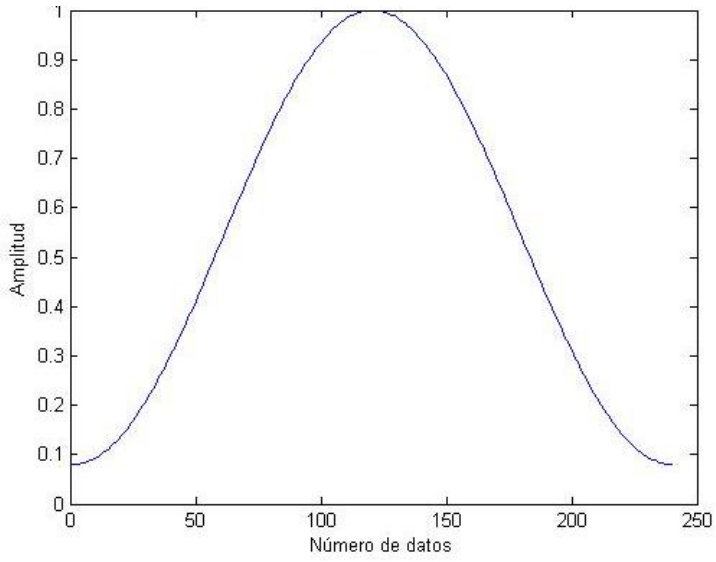


Figura. 2.15 Representación gráfica de la ventana Hamming

En este tipo de ventana se ve atenuada la distorsión producida por las discontinuidades. Además elimina los problemas causados por los cambios rápidos de la señal en los extremos de cada trama de voz. Por esta misma razón la segmentación se aplica con un desplazamiento inicial (para conseguir transiciones suaves entre tramas). En la práctica es deseable normalizar la ventana para que la potencia de la señal sea aproximadamente igual a la potencia de la señal antes del enventanado.

4. Ventana Blackman: se define mediante la ecuación 2. 11

$$w[k] = 0.42 - 0.5cos\left(2\pi\frac{k-1}{n-1}\right) + 0.08cos\left(4\pi\frac{k-1}{n-1}\right), \quad k = 1, \dots, n$$ Ecuación 2. 11

La forma de una ventana de Hamming se observa en la figura 2.16

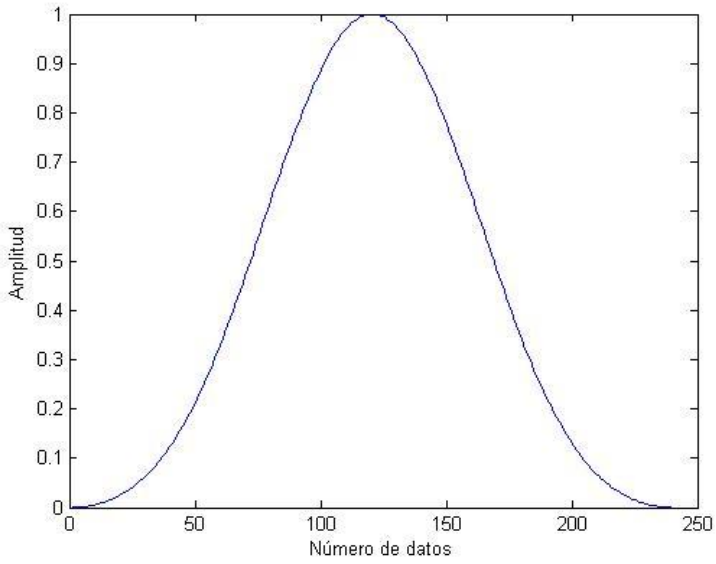


Figura. 2.16 Representación gráfica de la ventana Blackman

Los efectos en el dominio del tiempo para esta ventana no son muy diferentes de los anteriores, los efectos en dominio de la frecuencia son los que hay que tener especial cuidado.

Para notar la diferencia en la eficiencia de cada una de las ventanas se hace la siguiente comparación:

Con respecto al parámetro de amplitud todas las ventanas presentan un pequeño error, aunque este error no es muy notorio. En cuanto a otros parámetros como el ancho de banda del lóbulo principal, se puede decir que la ventana rectangular y hamming tienen un lóbulo principal muy definido, pero la atenuación de frecuencias parasitas no es tan eficiente como el de otras ventanas. Estas frecuencias parasitas son los lóbulos secundarios. Al comparar, las otras dos ventanas restantes; Hanning y Blackman vamos a encontrar similitud en cuanto a atenuación de frecuencias parasitas, amplitud y ancho de banda del lóbulo principal. En el caso de la ventana Blackman se observa que la aparición de frecuencias parasitas es menor pero tenemos un ancho de banda mayor para cada

pulso de frecuencia, es por ello que la ventana que mejor se acopla a los requerimientos del sistema de reconocimiento de voz es la ventana de Hanning, debido a que esta ventana hace disminuir la amplitud de la señal cerca de los bordes ayudando a eliminar las discontinuidades. La figura 2.17 muestra una comparación entre las técnicas de ventaneo.

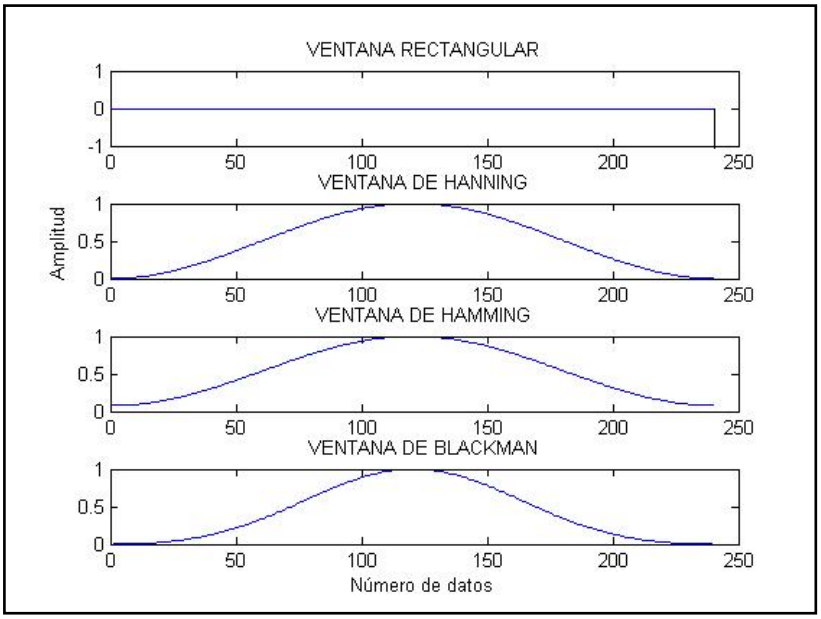


Figura. 2.17 Comparación entre técnicas de ventaneo

2.3.3 Análisis de características de la señal

En el reconocimiento del habla, la señal de voz pre-procesada se 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, ya que 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:

- Análisis de predicción lineal (LPC)

- Análisis cepstral

Durante el desarrollo del programa de reconocimiento de voz la técnica que se ha empleado es el análisis cepstral, por ello centraremos nuestra investigación en este tema.

2.3.3.1 Cepstrum

Como se sabe los sonidos de la voz se pueden representar mediante un espectrograma, que indica las componentes frecuenciales de la señal de voz. Es así como el espectro nos proporciona información acerca de los parámetros del modelo de producción de voz, tanto de la excitación como del filtro que representa el tracto vocal.

El cepstrum es una transformación común que se utiliza para obtener información de señal de voz de una persona. Se puede utilizar para separar la señal de excitación (que contiene las palabras y el tono) y la función de transferencia (que contiene la calidad de la voz). Es similar a un codificador de voz LPC en sus aplicaciones, pero utilizando el cepstrum como un analizador espectral se realiza un proceso completamente diferente y mejorado. La figura 2.18 muestra una comparación ente la técnica cepstrum y LPC. (Kinnunen, 2011)

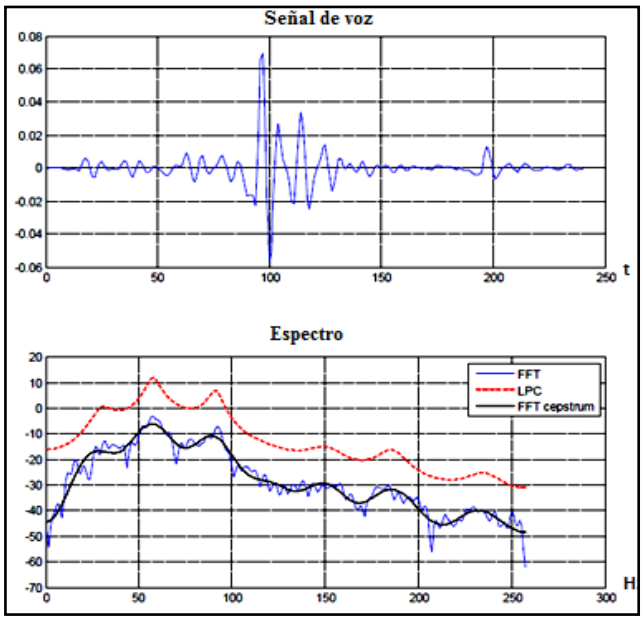


Figura. 2.18 Comparación del Cepstrum con un modelo de predicción lineal

El cepstrum representa una transformación de la señal de voz en base a dos propiedades importantes:

- Las señales componentes de la señal de voz serán separadas en el cepstrum.
- Las señales componentes serán linealmente combinadas en el cepstrum.

El cepstrum real (RC) de una señal de voz está definido por la ecuación 2.12.

$$C_s(n) = \tau^{-1}\{log|\tau\{S(n)\}|\} \quad y \quad C_s(w) = log|S(w)| \quad \text{Ecuación 2.12}$$

Donde $\tau\{.\}$ denota la Transformada de Fourier en Tiempo Discreto (DTFT). La fórmula anterior indica que el cepstrum de la señal se obtiene mediante el logaritmo de la señal de voz en tiempo discreto. El cepstrum se encuentra en el dominio de la frecuencia pero puede ser regresado al dominio del tiempo mediante la aplicación de la Transformada Inversa de Fourier en Tiempo Discreto (IDTFT).

El proceso de generación de la voz depende de dos factores básicos, uno de baja frecuencia llamado excitación $E(w)$, y uno de alta frecuencia generado por el sistema del tracto vocal $\theta(w)$. La convolución de estos dos términos genera la señal de voz como se muestra en la ecuación 2.13.

$$S(w) = E(w)\theta(w) \qquad \text{Ecuación 2.13}$$

Aplicando la definición de cepstrum se obtiene la ecuación 2. 14:

$$C_s(w) = log|S(w)|$$

$$C_s(w) = log|E(w)\theta(w)|$$

$$C_s(w) = log|E(w)| + log|\theta(w)|$$

$$C_s(w) = C_e(w) + C_\theta(w) \qquad \text{Ecuación 2.14}$$

Ahora que se encuentra en un dominio lineal, estamos libres para aplicar técnicas lineales a la nueva señal $C_s(w)$. Este es el objetivo de hallar el cepstrum de la señal, conseguir dos partes aditivas de la señal original a las cuales se les pueda aplicar un operador lineal, en este caso, la Transformada de Fourier.

Si se desea regresar al dominio del "Tiempo", lo que se debe hacer es aplicar la IDTFT como se indicó anteriormente. El nuevo dominio donde se encuentra la señal se llama quefrency aunque en realidad representa un dominio del tiempo. En este eje de tiempo, las componentes con más rápidas variaciones serán colocadas en los valores más altos del eje, y las que tengan variaciones lentas serán colocadas en los valores más bajos del eje.

La figura 2.19 resume el proceso de obtención del cepstrum de una señal.

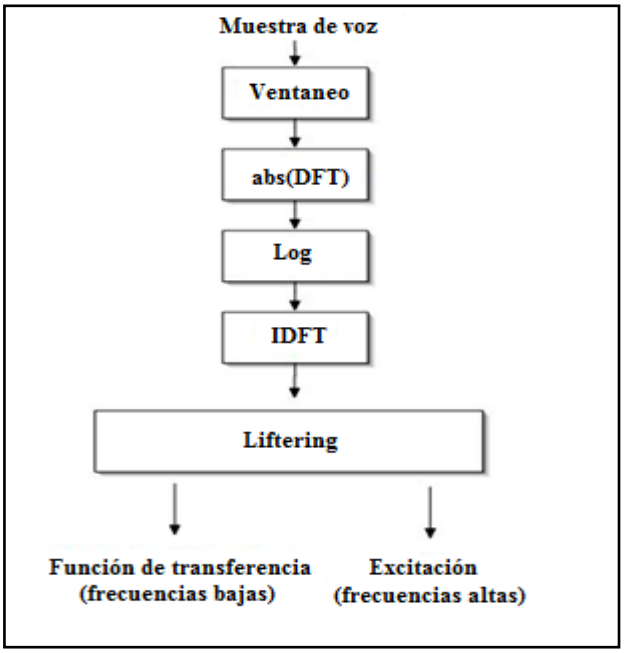


Figura. 2.19 Diagrama de bloques del cálculo del cepstrum de una señal

El esquema de la figura 2.20 resume los cambios que sufre la señal durante el proceso de cálculo del cepstrum.

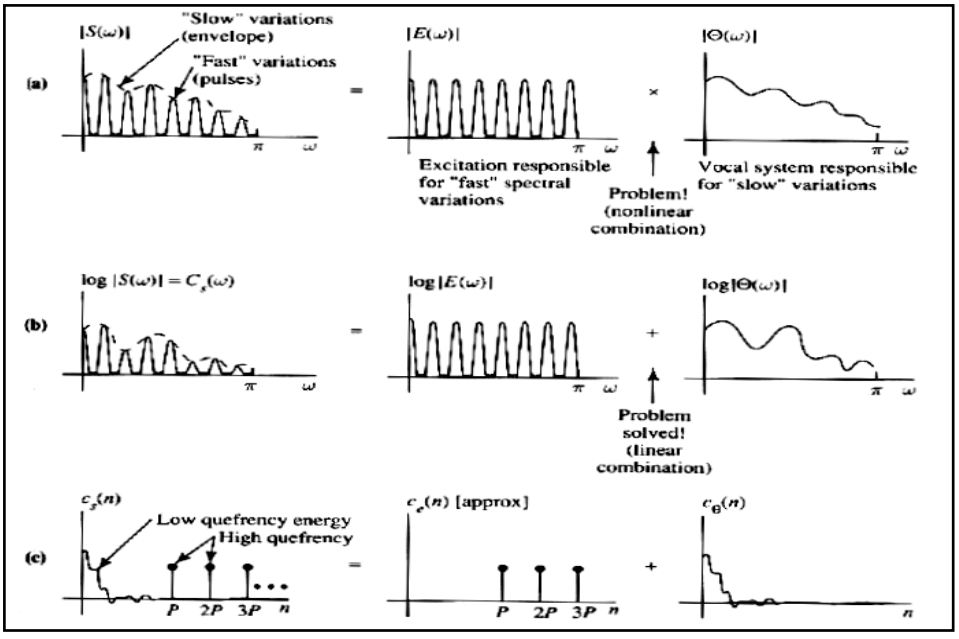


Figura. 2.20 Señal de voz durante el cálculo Cepstrum

Como se observa en la imagen, la señal de voz original está formada por la convolución de la señal de excitación y la correspondiente al sistema vocal. Esta convolución se convierte en una combinación lineal al aplicarse el logaritmo a la señal. Finalmente al regresar a la señal al dominio del "tiempo" (llamado generalmente dominio quefrency) se puede aplicar un filtro que permite pasar la parte más baja de la señal (liftering). Esta parte representa la señal de voz debida únicamente al sistema o tracto vocal. A partir de esta señal se pueden obtener los coeficientes cepstrales mostrados en la figura 2.21, que serán usados para el sistema de reconocimiento de voz.

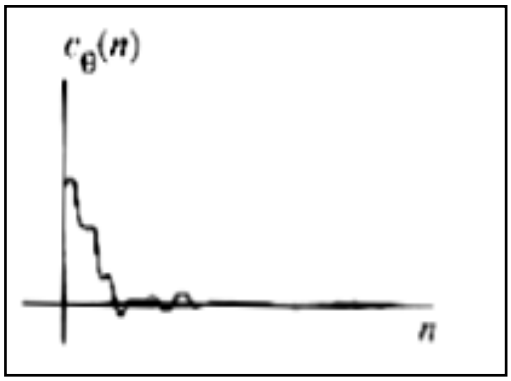


Figura. 2.21 Señal de voz característica debida al tracto vocal

La razón principal para utilizar los coeficientes cepstrales es que tienen la ventaja adicional que uno puede derivar 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.

2.3.4 Medida de 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. En el algoritmo de reconocimiento en MATLAB se recomienda utilizar una distancia Euclidiana (Velásquez, 2008), definida del siguiente modo: por ejemplo si *fi* y *fi',* con *i=0, 1, 2,..., D* son las componentes de dos vectores característicos *f* y *f'*, puede definirse la ecuación 2.15.

(Palacios, 2010)

$$d = \sqrt{\sum_{i=1}^{D} \left| f_i - f_{i'} \right|^2} \qquad \text{Ecuación 2.15}$$

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

CAPÍTULO 3

DESARROLLO DE LA APLICACIÓN

3.1 INTERFAZ MATLAB-ARIA[6]

Este capítulo hace referencia al uso del software ARIA y MATLAB, los cuales son los dos programas más importantes durante el desarrollo de la capa adaptadora.

La interfaz de MATLAB-ARIA es una aplicación escrita en lenguaje C++, que permite llamar a muchos comandos y acciones de ARIA desde MATLAB, ya que fueron construidos en funciones de MATLAB. Los programas escritos en código C++ y que se pueden llamar desde MATLAB se denominan Archivos-MEX. Estos Archivos-MEX poseen subrutinas que MATLAB automáticamente puede cargar y ejecutar.

Con el fin de determinar el mejor diseño y ejecución de la capa adaptadora, se ha profundizado en el estudio de la documentación de ARIA, MATLAB y API (Interfaz de programación de aplicaciones). Este esfuerzo también fue hecho para asegurarse de que la API resultante sea bastante simple e intuitiva para los nuevos usuarios sin necesidad de un entrenamiento extenso.

[6] ARIA and Matlab Integration With Applications

La figura 3.1 muestra el esquema de cómo la interfaz de MATLAB-ARIA interactúa con MATLAB, el entorno de simulación MobileSim y con el robot Pioneer P3-DX.

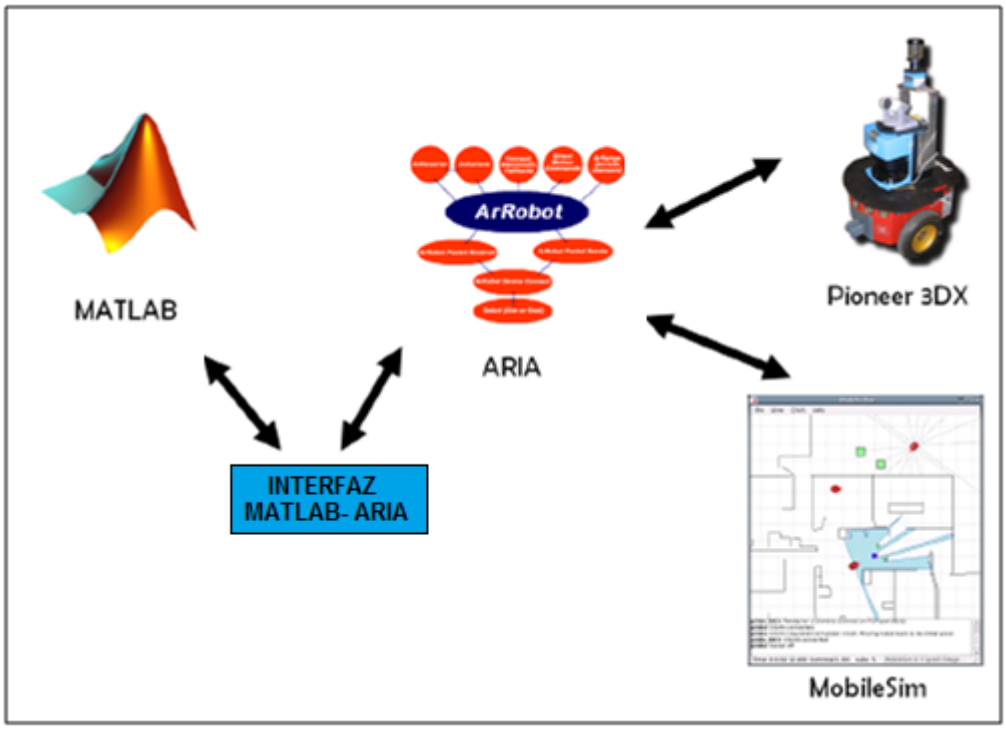


Figura. 3.1. Funcionamiento de la interfaz MATLAB-ARIA

3.1.1 Descripción de la interfaz MATLAB-ARIA

La compuerta de comunicación entre los software MATLAB y ARIA está escrita en lenguaje C++ desde Microsoft Visual Studio.NET 2003 y ha sido compilado en MATLAB R2009a con el propósito de generar un archivo MEX llamado ariainterface.mex. Se han implementado las funciones básicas de movimiento del robot pero la interfaz está sujeta a modificaciones con el objetivo de ampliar la funcionalidad del robot.

La función MEX es llamada desde el entorno de MATLAB como se muestra a continuación:

ariainterface(argumento1, argumento2)

Es decir se envían uno o dos argumentos dependiendo de cada función. Por lo tanto:

Nrhs=1 o 2

El vector que contiene los datos enviados es prhs[]. Generalmente se emplea el primer elemento de este vector para enviar el dato del número de la función llamada, aunque existen funciones que a más de este dato requieren una especificación adicional.

Prhs[0]=Contiene el número de la función llamada
Prhs[1]=Contiene el valor de la velocidad, aceleración, etc.

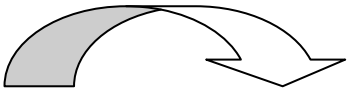
Por ejemplo si desde el entorno de MATLAB se desea ordenar al robot girar hacia la derecha, con una velocidad de 5grados/s, se procede de la siguiente manera:

ariainterface(4,5)

Donde 4 corresponde al número de función implementada en el código C++, en este caso es la función setRotVel, y 5 es el valor de la velocidad deseada en grados/s. Entonces tenemos:

Nrhs	=2	Número de datos de salida
Prhs[0]	=4	Función 4, setRotVel.
Prhs[1]	=5	Velocidad 5º/s

La tabla 3.1 muestra la función implementada en MATLAB, la llamada a la interfaz MATLAB-ARIA desde esa función y su traducción a una función de ARIA:



FUNCIÒN DE MATLAB	INTERFAZ MATLAB-ARIA	FUNCIÓN ARIA	DESCRIPCIÓN
robotInit()	ariainterface(0)	myRobotInit()	Esta función inicializa la conexión con el robot ya sea en el simulador o el robot físico.
shutdown()	ariainterface(1)	myShutdown()	Esta función apaga el robot y lo desconecta ya sea en el simulador o el robot físico
setVel(veltrans)	ariainterface(2,veltrans)	robot->setVel(*firstArg)	Esta función establece la velocidad transversal del robot en mm/s, siendo la velocidad máxima de 1200mm/s
stop()	ariainterface(3)	robot->stop()	Esta función detiene el movimiento del robot
setRotVel(velrot)	ariainterface(4,Velrot)	robot->setRotVel(*firstArg)	Esta función establece la velocidad rotacional deseada del robot en grados/s, siendo la velocidad máxima de rotación de 300 grados/s
setTransAccel(acel)	ariainterface(5,acel)	robot->setTransAccel (*firstArg)	Esta función establece la aceleración de traslación deseada del robot en grados/s^2. Max. 2000 grados/s^2
setTransDecel(desacel)	ariainterface(6,desacel)	robot->setTransDecel (*firstArg)	Esta función establece la desaceleración de traslación deseada del robot en grados/s^2. Max. 2000 grados/s^2

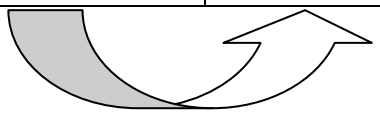


Tabla. 3.1Funciones de enlace de la interfaz MATLAB-ARIA

3.1.2 Guía de Compilación

El proceso de compilación debe realizarse desde una plataforma donde pueda operar ARIA. En este caso se empleará una PC de 32 bits, con sistema operativo Windows XP.

La compilación de la interfaz puede realizarse de dos formas:

- Desde MATLAB
- Desde Visual Studio

3.1.2.1 Compilación desde MATLAB[7]

Para comenzar con la compilación es necesario determinar qué versión de MATLAB puede construir funciones MEX y qué compilador requiere para ello. La tabla 3.2 muestra las características de una versión actualizada de MATLAB (MathWorks, 2013).

Compilador	**Versión**	**MATLAB 7.8 (MEX, Engine, MAT-file Access)**	**Compilador MATLAB 4.10**	**MATLAB Builder NE 3.0.1**	**MATLAB Builder EX 1.2.12**	**MATLAB Builder JA 2.0.3**
Intel C++	9.1	✓				
Intel Visual Fortran	10.1	✓				
	9.1	✓				
Lcc - win32	2.4.1	✓	✓			
Microsoft .NET Framework SDK	2.0			✓		
Microsoft	9.0	✓	✓	✓	✓	

[7] Math Works. Supported and Compatible Compilers – Release 2009b

Visual C++ 2008 SP1	Prof. Edition					
Microsoft Visual C++ 2008	9.0 Express Edition	✓	✓	✓	✓	
Microsoft Visual C++ 2005 SP1	8.0 Prof. Edition	✓	✓	✓	✓	
Microsoft Visual C++ .NET 2003	7.1 Prof. Edition	✓	✓	✓	✓	
Microsoft Visual C/C++	6.0 Prof. Edition	✓	✓	✓	✓	
Open Watcom	1.7	✓				
Sun Java Developmen t Kit (JDK)	1.5					✓

Tabla. 3.2 Compiladores compatibles soportados MATLAB R2009a

Al analizar la versión de 7.8.0 R2009a mostrada en la figura 3.2 se determina que sí es posible construir archivos MEX desde ésta versión de MATLAB.

Figura. 3.2 MATLAB 7.8.0 R2009a

Dado que la interfaz depende de ARIA, y las librerías de ARIA están pre-compiladas con Microsoft Visual C++ 7.1 (.NET 2003) como lo muestra la figura 3.3, se recomienda el uso de ésta versión para el proceso de compilación.

Figura. 3.3 Microsoft Visual C++ 7.1 (.NET 2003)

– Paso 1: Configurar el compilador que usará MATLAB para construir el archivo MEX. Para ello en el Command Window debe escribirse:

>> mex –setup

Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? ***y***

Select a compiler:

[1] Lcc-win32 C 2.4.1 in C:\ARCHIV~1\MATLAB\R2009a\sys\lcc

[2] Microsoft Visual C++ .NET 2003 in C:\Archivos de programa\Microsoft Visual Studio .NET 2003

[3] Open WATCOM C++ 1.7 in C:\WATCOM

[0] None

Compiler: ***2***

Please verify your choices:

Compiler: Microsoft Visual C++ .NET 2003

Location: C:\Archivos de programa\Microsoft Visual Studio .NET 2003

Are these correct [y]/n? ***y***

Done . . .

- Paso 2: Ejecutar la función makeMatlabAriaInterfaceVS2003, que básicamente carga en el programa los archivos include y las librerías de ARIA y de Visual Studio .NET2003. La parte fundamental de esta función está en el comando de compilación, implementado de la siguiente manera:

mex src\ariainterface.cpp, incFiles, libFiles

El archivo ariainterface.cpp contiene la programación de la interfaz MATLAB-ARIA en código C++, incFiles es la dirección de los archivos include y libFiles es la dirección de los archivos de librerías tanto de MATLAB como de ARIA.

3.1.3 Procedimiento para trabajar con la interfaz

La interfaz permite trabajar tanto con el simulador, como con el robot físico. Los pasos que se deben seguir son:

- Colocar el path en MATLAB
- Abrir el simulador MovileSim o conectar el cable de datos USB/SERIAL al robot.
- Conexión e inicialización

COLOCAR EL PATH EN MATLAB

Para que ARIA pueda ser llamada desde MATLAB mientras se esté trabajando desde un directorio diferente, la carpeta de la interfaz MATLAB-ARIA debe ser agregada en el path de MATLAB.

Para cambiar el path, abra el cuadro de diálogo **Set Path** desde:

->Archivo->Set Path, como se muestra en la figura 3.4

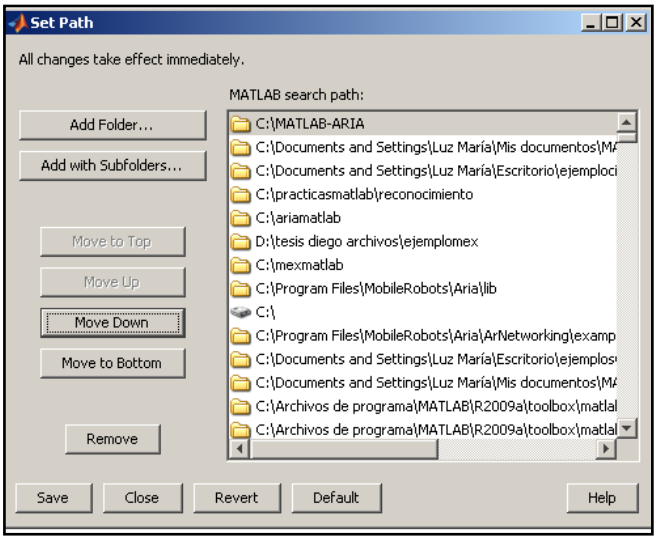


Figura. 3.4 Configuración del Path de MATLAB

Luego agregue la carpeta MATLAB-ARIA al path, guarde y cierre la ventana.

ABRIR MOBILESIM O CONECTAR EL ROBOT

Luego de que el path haya sido colocado y MATLAB esté todavía abierto, abra MobileSim. La ventana debe ser similar a la que se muestra en la figura 3.5.

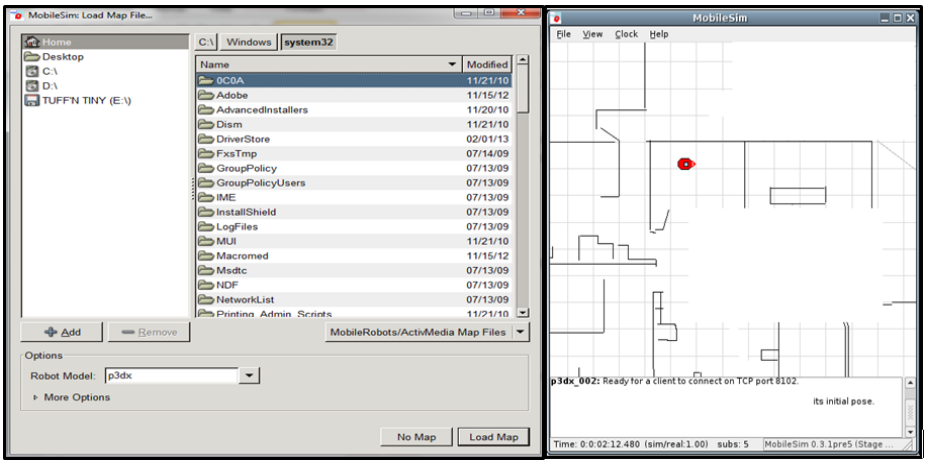


Figura. 3.5 Entorno de simulación MobileSim

En esta ventana podemos escoger la opción No Map para trabajar con el simulador sin usar un mapa de entorno o por el contrario podemos escoger la opción Load Map, en cuyo caso se debe buscar la ruta donde se encuentre el archivo del mapa.

En caso que se desee trabajar con el robot físico, debe conectarse el cable de datos (USB/RS232) desde la PC hasta el puerto serial del robot Pioneer P3-DX como lo muestra la figura 3.6.

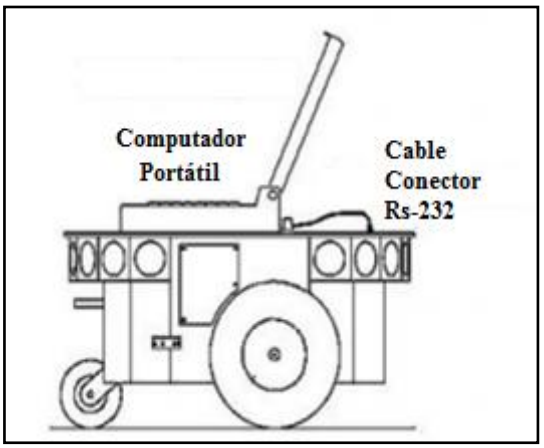


Figura. 3.6 Conexión PC-Robot Pioneer P3-DX

CONEXIÓN E INICIALIZACIÓN

Ahora en la ventana de comandos de MATLAB, escriba:

>>robotInit

Una vez ejecutado este comando se puede observar en el simulador que el robot está conectado o a su vez se escuchará cómo se encienden los motores en el robot físico. Este efecto se muestra en la figura 3.7 y 3.8 respectivamente.

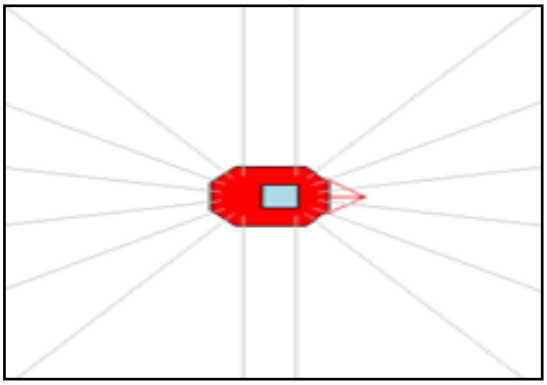


Figura. 3.7. Interfaz MATLAB-ARIA conectada con el simulador MobileSim

Figura. 3.8 Interfaz MATLAB-ARIA conectada con el robot físico

3.2 SISTEMA DE RECONOCIMIENTO DE VOZ

3.2.1 Diagrama de bloques del sistema[8]

El sistema de reconocimiento de voz se puede resumir en el diagrama esquemático mostrado en la figura 3.9.

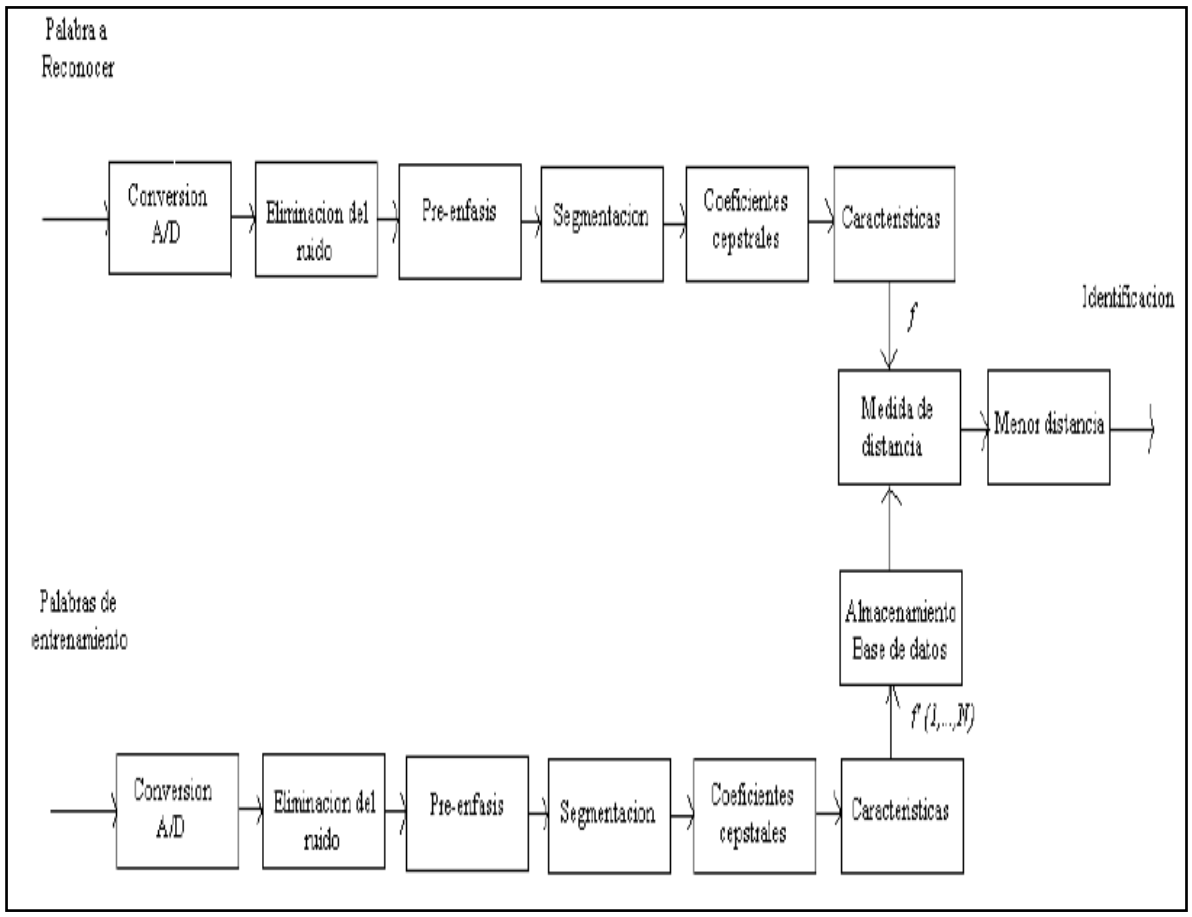


Figura. 3.9 Procesamiento de la señal de voz en el sistema de reconocimiento

Cada uno de los bloques descritos en el esquema anterior se encuentra implementado mediante una función de MATLAB. Cada función es llamada desde el

[8] Control de movimiento de un robot por medio de reconocimiento de voz.

programa principal tanto para la generación de la base de datos, así como para el preprocesamiento de la señal y la detección del comando ingresado.

3.2.2 Obtención de información mediante micrófono

La captura de voz se realiza en el PC bajo el software MATLAB, empleando un micrófono tipo diadema marca EB-330H. Teniendo en cuenta el teorema de muestreo y el ancho de banda en el cual se encuentra la señal de voz humana 100 - 3500Hz, la frecuencia de muestreo elegida fue de 8000 muestras/s, la cual es adecuada para la voz humana pero no para la reproducción musical. En la práctica permite reproducir señales con componentes de hasta 3,5 kHz. La obtención de la información mediante micrófono en MATLAB se realiza con la función wavrecord (t*Fs, Fs, ch), donde Fs es la frecuencia de muestreo, y t el número de segundos de la grabación. Debido a que el formato de audio wav es el formato propio de Windows, con 8 o 16 bits y con índices de muestreo de 11,025 kHz, 22,05 kHz o 44,1 kHz, para configurar la función wavrecord se ha escogido Fs=11025Hz, de manera que almacena 11025 datos cada segundo de grabación, lo cual puede ser considerado una buena calidad de sonido (Palacios, 2010). Posteriormente se emplearán para el análisis únicamente 8000 muestras por cada segundo de grabación. La señal obtenida es mono canal, que es el valor predeterminado en la función wavrecord, donde ch=1 se usa para mono y ch=2 para estéreo.

Por otro lado considerando la aplicación, la duración de la captura de la señal de voz es de 2 s, ya que el usuario recibe una señal sonora que le indica el momento en que va a empezar la grabación.

El comando empleado para realizar la obtención de la señal de voz mediante el micrófono es:

$$y = wavrecord(t * Fs, Fs, ch)$$

$$donde \quad t = 2, \qquad Fs = 11025, \qquad ch = 1$$

La figura 3.10 muestra una señal de voz grabada desde MATLAB usando el comando wavrecord.

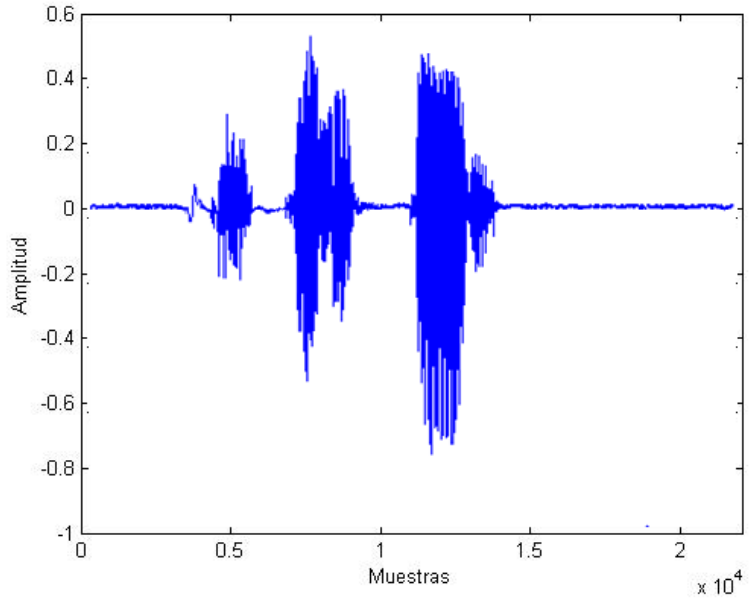


Figura. 3.10 Señal sin procesar obtenida del micrófono

El vector "y" que contiene la información de la palabra grabada tiene la siguiente dimensión:

[1 22050]

Se duplicó el número de muestras debido a que se grabó durante 2 segundos, y por cada segundo se tomaron 11025 datos, correspondiente a la frecuencia de muestreo de la función wavrecord de MATLAB.

3.2.3 Base de datos del programa

La base de datos consiste en muestras almacenadas en un archivo .mat de siete comandos distintos, todas estas muestras son tomadas de un mismo usuario y pueden por

lo tanto ser modificadas en cualquier instante. El conjunto de comandos implementados se muestra en la tabla 3.3.

COMANDO DE VOZ
Conectar
Fin
Adelante
Atrás
Paro
Giro uno
Giro dos

Tabla. 3.3 Palabras almacenadas en la base de datos del sistema

El proceso de generación de la base de datos se encuentra implementado en la función de MATLAB:

```
function [  ] = funBD(  )
```

Desde ésta instancia la palabra grabada se envía a una serie de funciones para ser preprocesada y mejorada antes de almacenarse como un vector de MATLAB y ser usada en el sistema de reconocimiento de voz.

3.2.4 Preprocesamiento

3.2.4.1 Muestreo y cuantificación

Como ya se dijo, se tomará una frecuencia de 8kHz, de esta manera se cumple con el teorema de Nyquist, tomando como base una frecuencia de la voz humana de hasta los 3.5 kHz

Es evidente los beneficios que presta realizara a esta señal una cuantificación vectorial en bloques de N muestras. En este tipo de cuantificación, el bloque de N muestras se trata como un vector N-dimensional.

3.2.4.2 Eliminación de ruido

Esta etapa de procesamiento de la señal se encuentra implementada mediante la función chop_silencio en MATLAB:

function [y] = chop_silencio(s)

Durante este proceso, la señal de voz se analiza por partes. Primero se forman segmentos de 10ms cada uno, es decir tramos de 80 muestras. Luego se obtiene el promedio energético de la señal entera y el de cada segmento considerando que la energía entre (N1, N2) de una señal discreta está dada por la ecuación 3.1.

$$E[x[n]]_{N_1 \to N_2} = \sum_{n=N_1}^{N_2} \left| x[n] \right|^2$$

Ecuación 3.1

Finalmente se establece un valor umbral correspondiente al 20% del promedio energético de la señal entera. De esta manera si el promedio del segmento analizado es menor que el valor umbral, se descarta como espacio en blanco y no se almacena, caso contrario se guarda en un vector que al final del proceso es retornado para continuar con el análisis. La figura 3.11 muestra la señal de voz con intervalos de silencio y ruido eliminados.

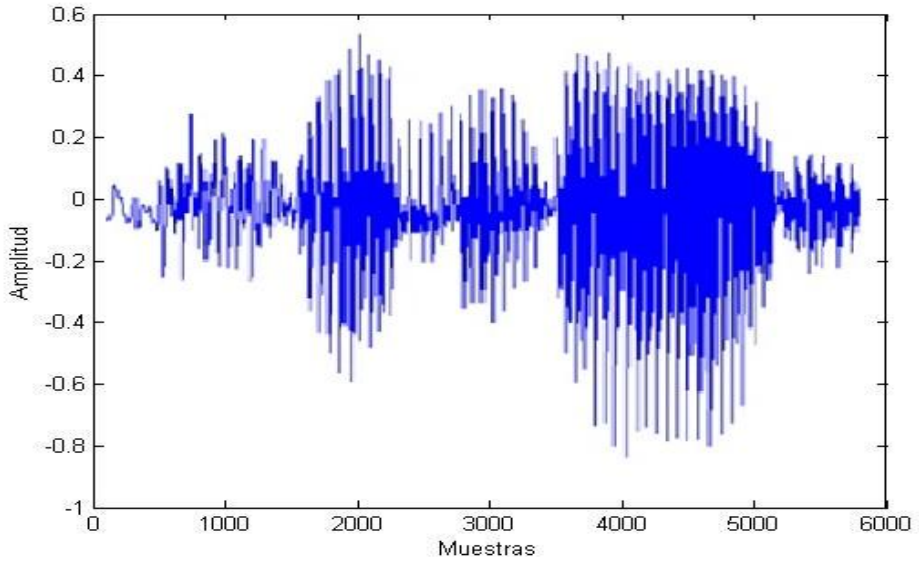


Figura. 3.11 Señal con ruido y silencio eliminado

Como se observa en la figura anterior, el número de datos, para la señal que se está analizando en este caso, se ha reducido de 22050 a solamente 5921 muestras, el resto ha sido eliminado como espacio en blanco. Para el caso particular de la palabra “Conectar” que estamos tratando como ejemplo, la dimensión del vector que contiene la señal de voz ahora es de:

[1 5921]

3.2.4.3 Filtro de Pre-énfasis

Esta etapa de procesamiento de la señal se encuentra implementada mediante la función énfasis en MATLAB:

```
function [y]=enfasis(x)
```

La figura 3.12 muestra la señal de voz en su tercera etapa de preprocesamiento, luego de pasar por el filtro de preénfasis.

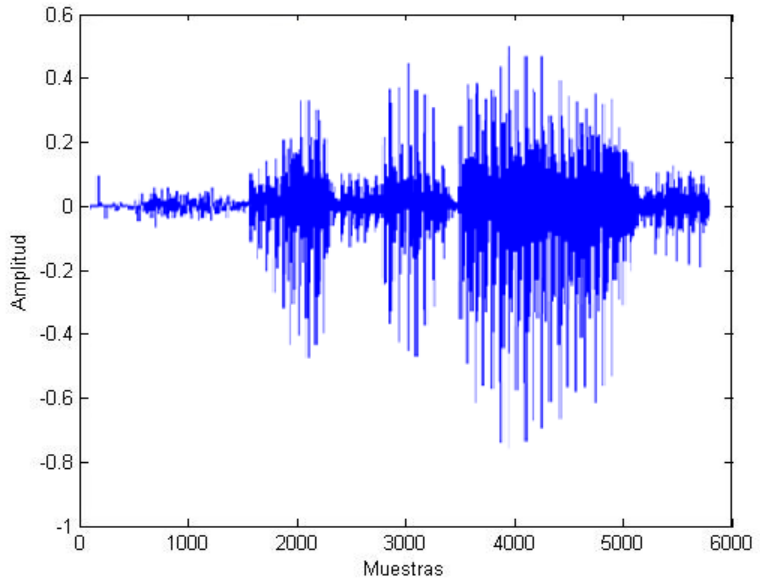


Figura. 3.12 Señal luego de la etapa de preénfasis

El filtro de preénfasis permite pasar frecuencias por encima de los 100 Hz, debido a que se muestrea voz humana. La dimensión del vector luego de pasar por el filtro se mantiene:

```
[ 1        5921]
```

3.2.4.4 Segmentación

Esta etapa de procesamiento de la señal se encuentra implementada mediante la función segmentos en MATLAB:

function [segs]=segmentos(y)

Durante la segmentación, la señal de voz es dividida primero en tramos de 10 ms (80 datos). Estos tramos se agrupan cada 30ms (240 datos) con un desplazamiento inicial de 10ms. Cada uno de estos segmentos se almacena en una fila diferente de un vector. La tabla 3.4 resume las características de segmentación.

Parámetro	Valor
N número de muestras en el segmento de análisis.	240 (30*ms*)
M número de muestras entre cada segmento.	80 (10*ms*)

Tabla. 3.4. Parámetros de segmentación del espectro de voz

Los datos se recopilan como muestra la figura 3.13.

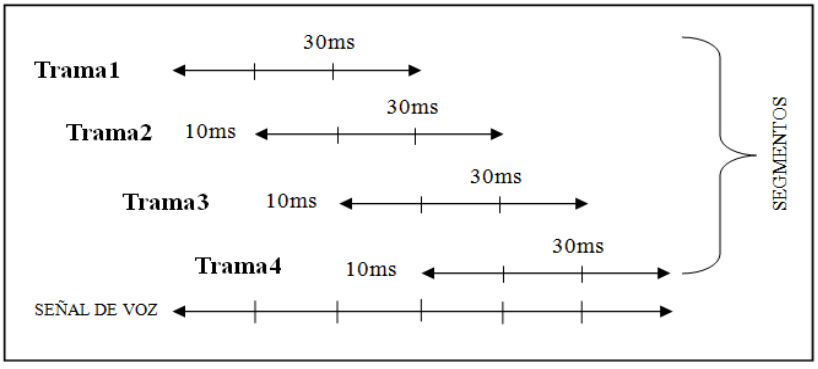


Figura. 3.13 Segmentación de la señal de voz

La primera trama de voz es la trama 1, que comienza desde el primer dato. La segunda trama no comenzará inmediatamente, sino que estará desplazada 10 mseg. respecto del comienzo de la trama anterior, y así sucesivamente. En la figura anterior, la punta de la flecha y su longitud indica el comienzo y duración de la trama. Es por esto que se habla de duración de las tramas y desplazamiento.

Antes de que la matriz segs sea retornada cada una de las tramas de 30ms pasa por una ventana de Hamming de 30 ms también. Esta ventana es creada mediante la función de MATLAB:

hamming(240)

La ventana de hamming de 240 puntos se muestra en la figura 3.14.

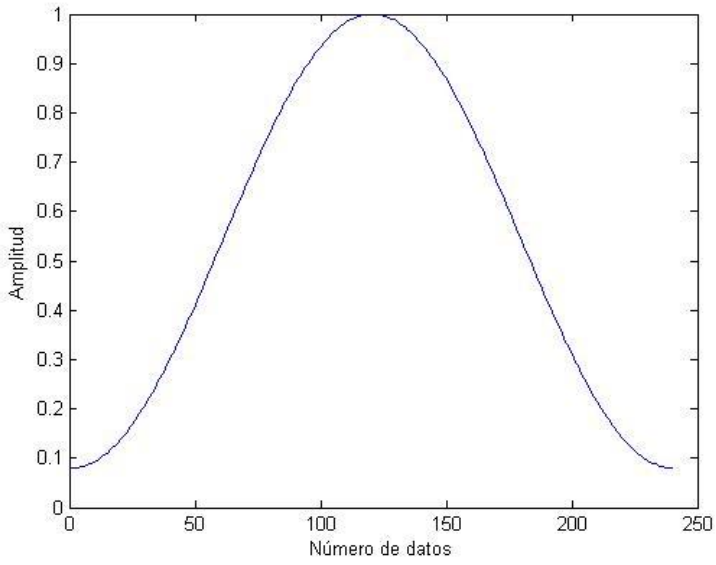


Figura. 3.14 Ventana de Hamming de 240 puntos

Luego de pasar por la ventana de Hamming, los segmentos se ven como se muestra en la figura 3.15.

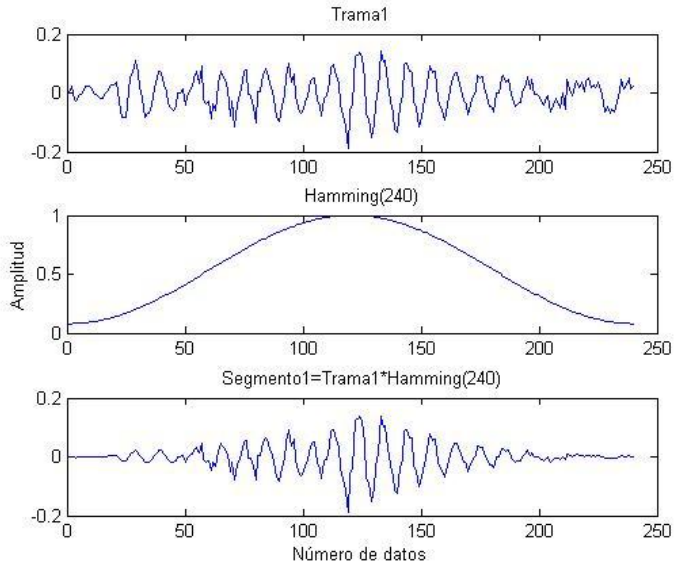


Figura. 3.15 Segmentación y ventaneo de la señal de voz

Para el caso de la señal que estamos analizando en éste ejemplo, se generan 72 segmentos de 240 datos cada uno. El vector a la salida de esta función tiene la dimensión:

[72 240]

3.2.5 Análisis de características de la señal

3.2.5.1 Cepstrum

Esta etapa de procesamiento de la señal se encuentra implementada mediante la función cepstrum en MATLAB:

function [c] = cepstrum(segs)

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 el capítulo anterior.

```
[xhat,yhat] = rceps(y)
```

Como ya se dijo se analizan todas las tramas, pero se consideran solo los 10 primeros coeficientes cepstrales de cada segmento correspondientes a los elementos característicos del tracto vocal, los elementos restantes de la matriz cepstrum no se emplean debido a que corresponden a la excitación de la señal de voz. El vector que resulta de esta etapa del preprocesamiento tiene la siguiente dimensión:

```
[72    10]
```

Por ejemplo, en el caso del segmento o trama número 1, se extrajeron los siguientes coeficientes cepstrales:

```
[ -0.6569, -0.5439, 0.3791, -0.2637 ,  0.2153 ,  0.0426 ,  0.0731 , 0.0554, 0.1481  ,  0.0242]
```

Cada conjunto de coeficientes cepstrales resulta del análisis del segmento correspondiente, recordando que cada segmento se tomó con un desplazamiento inicial de 10 ms del anterior.

En la figura 3.16 podemos analizar la variación de este primer grupo de coeficientes cepstrales correspondientes al segmento número 1:

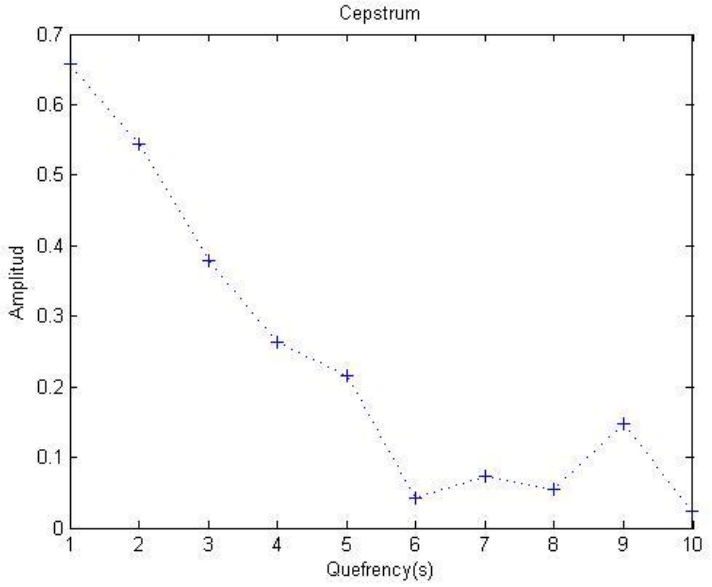


Figura. 3.16 Coeficientes cepstrales para un segmento

Para una mejor comprensión la gráfica 3.17 muestra todos los segmentos de la señal de voz representados mediante sus coeficientes cepstrales:

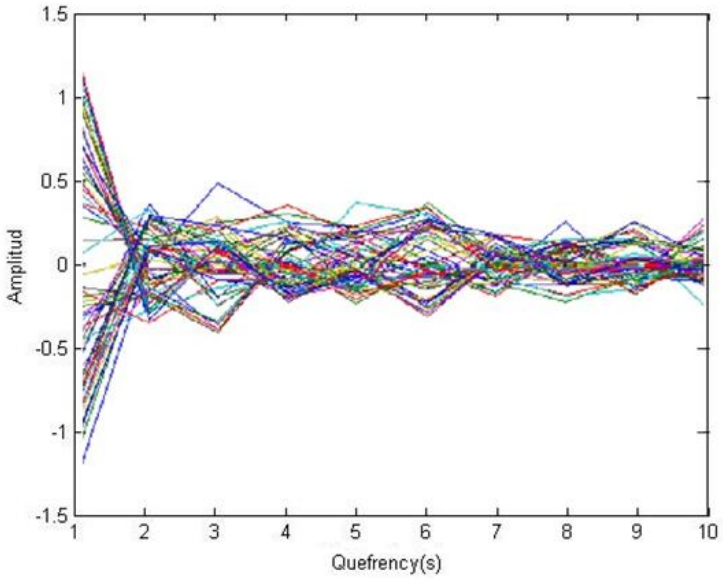


Figura. 3.17 Señal completa de voz representada mediante sus coeficientes cepstrales

3.2.6 Medida de distancia

El proceso descrito anteriormente es aplicado tanto a las palabras que se guardan en la base de datos como a las grabadas cuando el usuario se encuentra en el mando por voz. A continuación la matriz de coeficientes cepstrum de la palabra grabada se compara con la matriz cepstrum de cada palabra almacenada en la base de datos.

La función de MATLAB empleada para realizar este proceso es:

d=distancia (F,rec)

Donde F es la matriz de características correspondiente a una de las palabras de la base de datos y rec la matriz de características de la palabra grabada.

Durante el proceso, primero se miden los tamaños de cada una de las matrices y se realizan comparaciones para determinar cuál será la matriz guía y cual la esclava. Esto se realiza con el propósito de trabajar siempre en función de la matriz más pequeña (matriz guía), para evitar desbordamientos al acceder a los datos. Una vez determinado lo anterior, se mide la distancia Euclidiana entre puntos correspondientes de las dos matrices, para ello se ocupa la ecuación 3.2:

$$d = \sqrt{\sum_{i=1}^{D} |f_i - f'_i|^2} \qquad \text{Ecuación 3.2}$$

Donde f_i representa un dato de la posición i de la matriz de características de la palabra de la base de datos y f'_i el dato de la posición i de la matriz de características de la palabra grabada.

Al final del proceso se retorna una medida de distancia total, resultado de comparar las dos matrices. Desde el programa principal se repite el proceso de comparación varias veces para determinar la distancia con cada una de las palabras de la base de datos. Al generar un vector que almacene las medidas de distancias, podemos identificar la menor distancia obtenida y de ésta manera deducir si se ha acertado o no en el algoritmo, y cuál fue la palabra reconocida.

3.3 DESARROLLO DE LA INTERFAZ DE USUARIO

3.3.1 Descripción de la interfaz de usuario

Se ha desarrollado mediante una recopilación de dos programas, que una vez integrados, permiten controlar remotamente por voz el robot móvil Pioneer P3-Dx para el laboratorio de robótica de la Escuela Politécnica del Ejército. El entorno gráfico del sistema de Control Remoto por Voz del Robot Pioneer P3-DX es como se muestra en la figura 3.18:

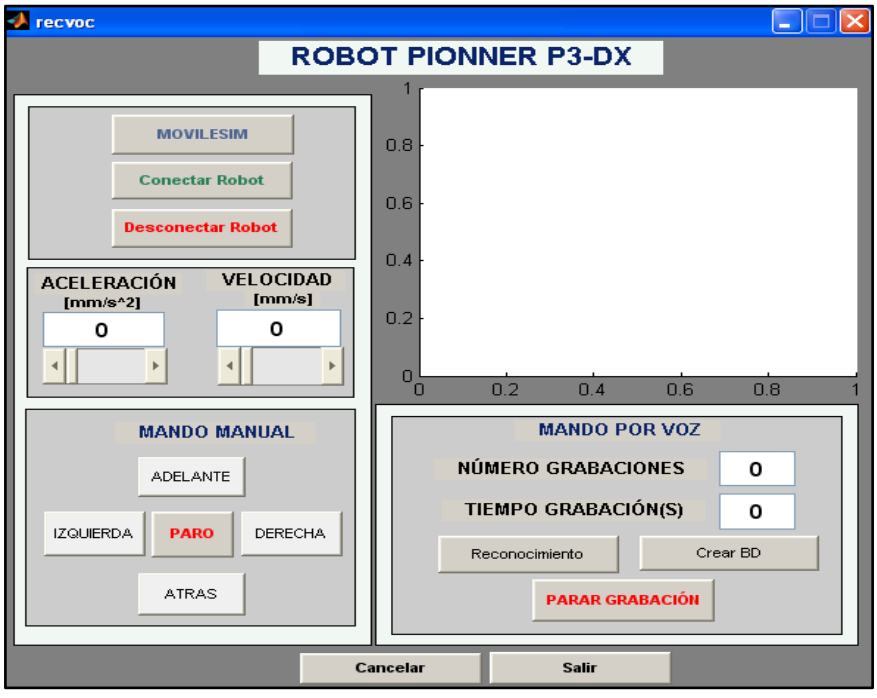


Figura. 3.18. Interfaz de la aplicación de CONTROL REMOTO POR VOZ desarrollada en MATLAB

3.3.2 Paneles de mando

Podemos diferenciar en esta interfaz tres paneles: configuración, mando manual y mando por voz.

3.3.2.1 Panel de Configuración

El panel de configuración del sistema de reconocimiento de voz se muestra en la figura 3.19.

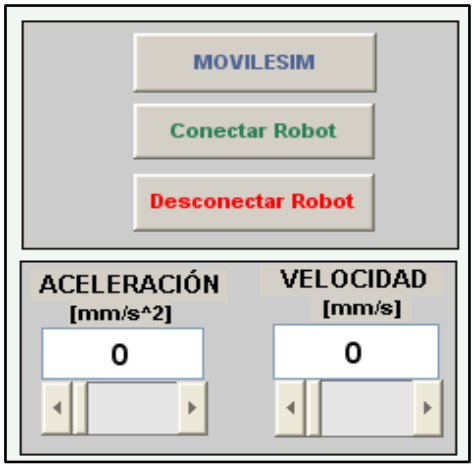


Figura. 3.19. Panel de configuración

Este panel le permite al usuario configurar parámetros básicos de movimiento del robot. Estos parámetros se describen en la tabla 3.5.

NOMBRE	ILUSTRACIÓN	DESCRIPCIÓN
Botón MOBILESIM		Este botón le permite al usuario abrir el simulador MOBILESIM del robot

Botón Conectar Robot		Este botón establece la conexión inicial con el robot previo a su uso en modo manual
Botón Desconectar Robot		Este botón finaliza la conexión con el robot, cuando se encuentra en modo manual
Slider de Aceleración		Este slider permite establecer la aceleración de movimiento del robot en un rango de 0 a 2000 mm/s^2
Slider de Velocidad		Este slider permite establecer la velocidad de movimiento del robot en un rango de 0 a 2000 mm/s2

Tabla. 3.5. Descripción del panel de configuración

3.3.2.2 Panel de Mando Manual

El panel de mando manual del sistema de reconocimiento de voz se muestra en la figura 3.20.

Figura. 3.20. Panel de mando manual

Este panel le permite al usuario controlar al robot en forma manual presionando cada uno de los botones de accionamiento que se describen en la tabla 3.6.

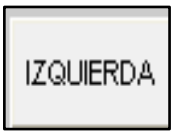
NOMBRE	ILUSTRACIÓN	DESCRIPCIÓN
Botón ADELANTE		Este botón le permite al usuario mover al robot hacia adelante con la velocidad y aceleración establecidas en el panel de configuración
Botón ATRÀS		Este botón le permite al usuario mover al robot hacia atrás con la velocidad y aceleración establecidas en el panel de configuración
Botón DERECHA		Este botón le permite al usuario mover al robot hacia la derecha
Botón IZQUIERDA		Este botón le permite al usuario mover al robot hacia la izquierda
Botón PARO		Este botón le permite al usuario parar el movimiento del robot

Tabla. 3.6. Descripción del panel de mando manual

3.3.2.3 Panel de Mando por Voz

El panel de Mando por Voz del sistema de reconocimiento se muestra en la figura 3.21.

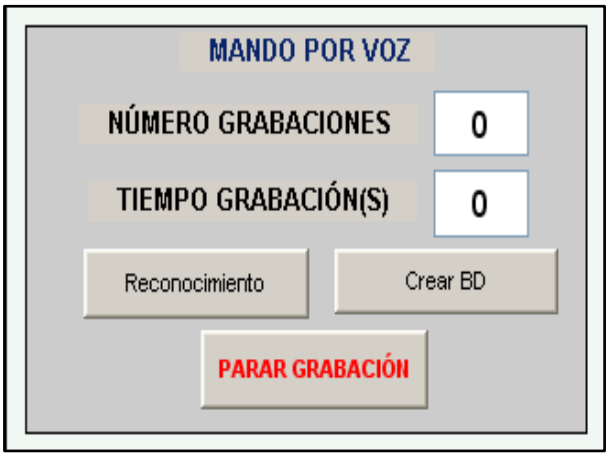


Figura. 3.21. Panel de mando por voz

Este panel le permite al usuario controlar remotamente por voz al robot fijando los parámetros descritos en la tabla 3.7.

NOMBRE	ILUSTRACIÓN	DESCRIPCIÓN
Número de Grabaciones		Esta entrada de texto le permite al usuario ingresar el número de ciclos de grabación que el mando por voz va a estar activo.
Tiempo de Grabación		Esta entrada de texto le permite al usuario ingresar el tiempo de cada ciclo de grabación.

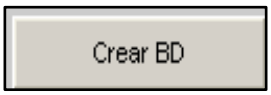
Botón de Reconocimiento		Este botón le permite al usuario activar el mando por voz. Si la activación es correcta se escucha la señal de voz "MANDO POR VOZ ACTIVADO"
Botón Crear Base de Datos		Este botón le permite al usuario guardar su voz en la base de datos, para que pueda comandar al robot.
Botón Parar Grabación		Este botón le permite al usuario parar la ejecución del mando por voz del robot. Si el comando se ejecuta correctamente se escucha la señal de voz "LA GRABACIÓN HA SIDO INTERRUMPIDA"

Tabla. 3.7 Descripción del panel de mando por voz

Adicionalmente se cuenta con dos botones en el panel principal y una ventana de visualización del espectro de la voz, estos elementos se describen en la tabla 3.8.

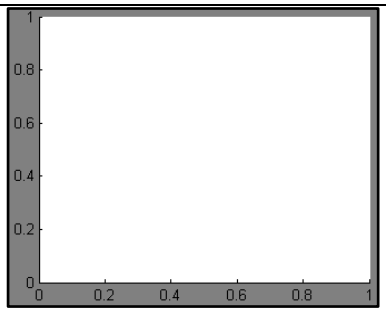
NOMBRE	ILUSTRACIÓN	DESCRIPCIÓN
Ventana Gráfica		Esta ventana le permite al usuario visualizar el espectro de su voz, sirve además de guía para saber que el sistema está en línea.
Botón Cancelar		Este botón le permite al usuario cancelar cualquier proceso que se esté ejecutando, inclusive la generación de la base de datos, luego de lo cual cierra automáticamente toda la aplicación.
Botón Salir		Este botón le permite al usuario cerrar la interfaz de usuario.

Tabla. 3.8. Descripción de los elementos del panel principal

3.3.3 Creación de la Base de Datos

El sistema de mando por voz del robot Pioneer P3-DX está diseñado para que cualquier persona pueda convertirse en operario y manejar al robot en cualquier momento.

Para ello se diseñó un botón llamado “Crear Base de Datos”, el cual se muestra en la figura 3.23.

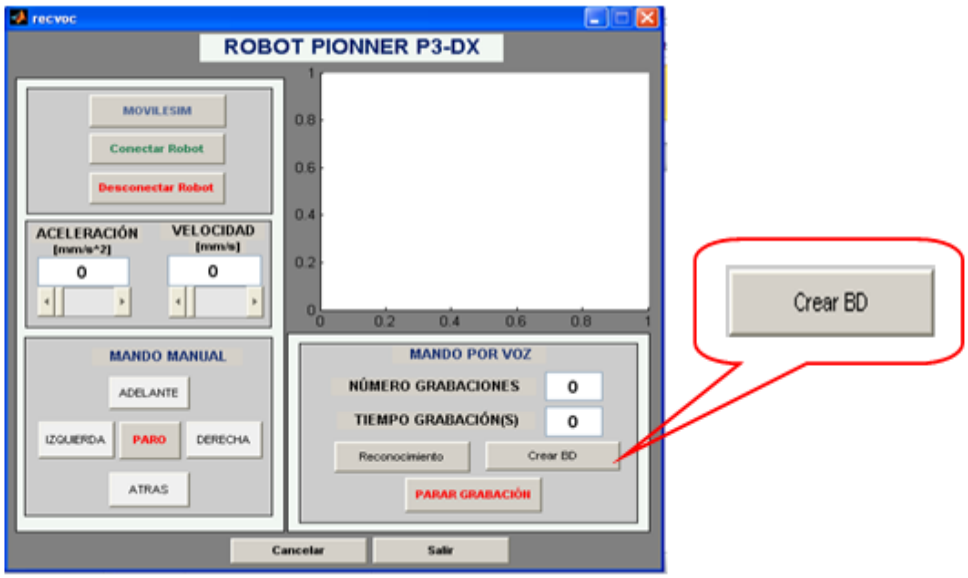
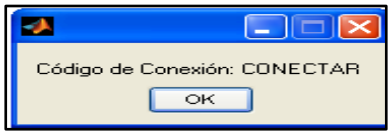


Figura. 3.22. Botón para la creación de la base de datos

Al presionar este botón, el sistema entra en un proceso de generación de su base de datos. El usuario comienza a visualizar las órdenes de grabación de cada uno de los comandos en cuadros de texto. Luego de realizarse este proceso, el sistema está listo para que el nuevo usuario pueda comandar el robot. La generación de la base de datos puede visualizarse en la tabla 3.9.

CUADROS DE TEXTO	ACCIÓN
	Aviso de advertencia para que el usuario esté listo para comenzar a grabar los comandos.
	Grabación en la base de datos del comando CONECTAR. La grabación está programada para 2 segundos
	Grabación en la base de datos del comando FIN. La grabación está programada para 2 segundos

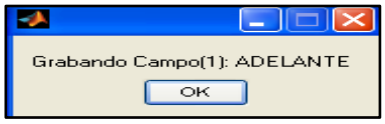
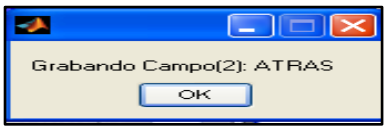
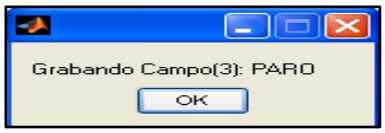
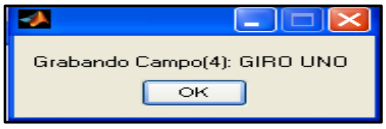
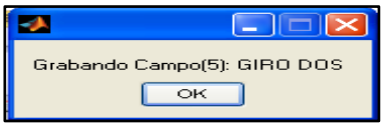
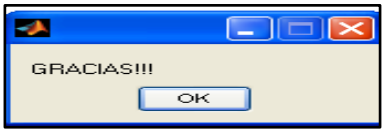
	Grabación en la base de datos del comando ADELANTE. La grabación está programada para 2 segundos
	Grabación en la base de datos del comando ATRÁS. La grabación está programada para 2 segundos
	Grabación en la base de datos del comando PARO. La grabación está programada para 2 segundos
	Grabación en la base de datos del comando GIRO UNO. La grabación está programada para 2 segundos
	Grabación en la base de datos del comando GIRO DOS. La grabación está programada para 2 segundos
	Al visualizar el mensaje GRACIAS, la creación de la base de datos finaliza.

Tabla. 3.9. Proceso de creación de la base de datos

3.3.4 Descripción de los Comandos de Voz

El usuario del sistema posee siete comandos de voz para controlar el robot, entre ellos encontramos Conectar, Fin, Adelante, Atrás, Paro, Giro uno y Giro dos. En la tabla 3.10 se describen las acciones que realiza cada uno de los comandos de voz cuando una palabra es reconocida.

COMANDO DE VOZ	ACCIÓN
Conectar	Establece la conexión entre la aplicación y el simulador MOBILESIM o el robot físico.
Fin	Desconecta la aplicación del simulador MOBILESIM o el robot físico
Adelante	El robot se mueve hacia adelante con la velocidad y aceleración establecidas en el panel de configuración
Atrás	El robot se mueve hacia atrás con la velocidad y aceleración establecidas en el panel de configuración
Paro	El robot se detiene.
Giro uno	El robot gira hacia la derecha.
Giro dos	El robot gira hacia la izquierda.

Tabla. 3.10. Descripción de los comandos de voz y sus acciones

CAPÍTULO 4

PRUEBAS Y RESULTADOS

Debido a que la plataforma Pioneer P3-DX es para ambientes interiores, las pruebas se llevaron a cabo en entornos interiores y estáticos es decir sin presencia de objetos en movimiento. Estas pruebas se efectuaron tanto con el simulador MOVILESIM como con el robot Pioneer P3-DX del laboratorio de robótica de la Escuela Politécnica del Ejército. La figura 4.1 y 4.2 muestran las pruebas de funcionamiento del sistema.

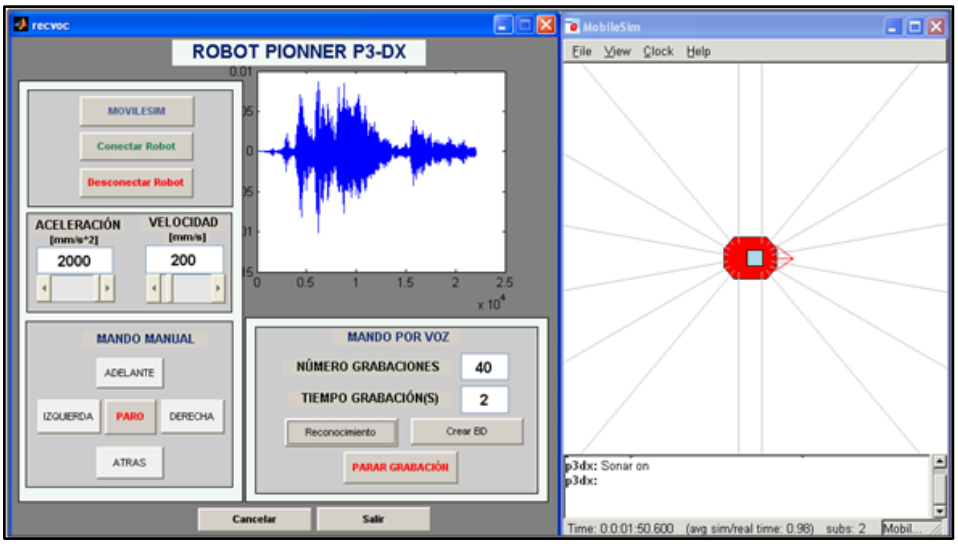


Figura. 4.1 Pruebas de funcionamiento PC - SIMULADOR MOVILESIM

Figura. 4.2 Pruebas de funcionamiento PC-ROBOT

4.1 DISEÑO DE PRUEBAS DE VALIDACIÓN

Las pruebas se realizaron en base a una taza de aciertos durante un ciclo de activación del mando por voz del robot. Esta taza de aciertos depende directamente de la similitud entre el espectro preprocesado de la señal grabada en tiempo real y la señal grabada en la base de datos.

El sistema de reconocimiento de voz identifica una palabra determinando la menor distancia entre las señales mencionadas anteriormente.

La tabla 4.1 muestra las distancias medidas durante un ciclo de activación del mando por voz para cada una de las palabras posibles:

Conectar	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	10.78	20.43	17.53	14.04	19.06	26.28	20.54

Fin	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	22.50	11.56	22.26	15.76	15.10	17.56	20.66

Adelante	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	16.67	19.80	10.54	12.96	16.43	29.36	24.24

Atrás	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	16.17	14.67	14.09	7.45	12.69	15.39	14.68

Paro	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	18.12	14.71	15.39	12.57	7.81	16.31	16.45

Giro Uno	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	25.50	16.76	31.64	15.99	17.9	14.53	23.43

Giro Dos	Conectar	Fin	Adelante	Atrás	Paro	Giro Uno	Giro Dos
	21.21	21.04	25.20	15.96	19.21	21.40	12.04

Tabla. 4.1 Matriz de distancias retornada al pronunciar cada palabra

Estos valores de distancia medidos varían dependiendo de varios factores. Los más importantes son el nivel de volumen del micrófono y la fuerza de voz con que se pronuncian las palabras. La tabla 4.2 muestra la medida de distancia promedio retornada para cada una de las palabras reconocidas correctamente, manteniendo un volumen del micrófono constante para cada prueba:

PALABRA A RECONOCER	PRUEBA 1	PRUEBA 2	PRUEBA 3	PROMEDIO
Conectar	10,79	11,26	12,37	11,5
Fin	11,89	12,11	11,96	12,0
Adelante	10.,64	10,68	11,15	10,9
Atrás	7,76	7,14	7.91	7,5
Paro	7,49	7,65	7,53	7,6
Giro uno	14,05	13,29	14,7	14,0
Giro dos	12,25	12,95	12,81	12,7

Tabla. 4.2 Distancia promedio obtenida para tres pruebas de reconocimiento

Los resultados mostrados en la tabla anterior permiten discriminar medidas de distancia por encima de los valores promedio. Es por ello que medidas mayores a 15 se asimilarán como palabras diferentes a las existentes en la base de datos y por ello serán rechazadas por el sistema, en cuyo caso se escucha la advertencia sonora: NO DETECTADO.

4.2 TABULACIÓN DE RESULTADOS

Considerando que la tasa de aciertos de una determinada configuración es la medida aritmética de la tasa de acierto de los ejemplos presentados. Para cada uno de los ejemplos, se considera que la tasa de acierto es la proporción entre el número de palabras reconocidas positivamente y el número de palabras consideradas en la evaluación del sistema. La fórmula empleada para el cálculo de la tasa de aciertos se muestra en la ecuación 4.1.

$$\text{tasa de aciertos}(\%) = \frac{\text{No.de aciertos}}{\text{No.de casos}} \text{x}100 \qquad \text{Ecuación 4.1}$$

En base a este concepto, para el experimento se consideraron 40 grabaciones con una duración de 2 segundos para cada una.

Las palabras a reconocer pueden ser mencionadas de manera aleatoria del conjunto de palabras almacenadas previamente en la base de datos del sistema.

En la tabla 4.3 se muestran los aciertos y desaciertos al pronunciar un comando de voz considerando que el operador dice una palabra en cualquier instante, sin prevenir que se esté realizando la grabación de la orden o no:

COMANDO	ACIERTOS	DESACIERTOS
Conectar	2	2
Fin	2	2
Adelante	2	2
Atrás	4	0
Paro	4	0
Giro Uno	2	2
Giro Dos	3	1
SUMA	**19**	**9**
	TOTAL CASOS	**28**

Tabla. 4.3 Resultados obtenidos para las pruebas iniciales

Los resultados de la tabla anterior se muestran en la figura 4.3.

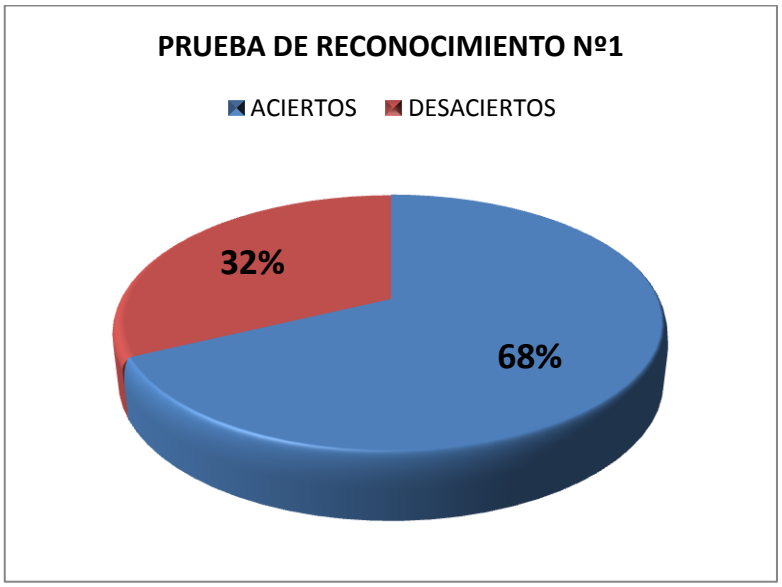


Figura. 4.3 Taza de aciertos para pruebas iniciales

El cálculo de la tasa de aciertos en esta configuración del sistema se muestra en la ecuación

$$tasa\ de\ aciertos(\%) = \frac{19}{28} x100$$

$$tasa\ de\ aciertos(\%) = 67.85\%$$

Como podemos notar la tasa de aciertos es relativamente baja, esto se debe a que el sistema de reconocimiento de voz es un proceso de ejecución continua y por esta razón el usuario puede pronunciar los comandos de voz en el momento en que él desee.

Inicialmente se propuso un sistema donde el usuario debía hablar justo en el momento en el cual se estaba realizando la grabación, pero en ocasiones éste momento era poco evidente y por ello ocurrían errores en los cuales las palabras se grababan incompletas y se enviaban comandos erróneos, como ocurrió en la prueba realizada anteriormente. De allí nació la necesidad de buscar una forma de anunciar al usuario que la grabación se está realizando. La solución más efectiva y que fue implementada en el sistema fue un pequeño sonido de alerta, con la cual el usuario sabe el instante en que inicia la grabación de una palabra y procede a decir el comando, evitando así grabar palabras incompletas.

En la tabla 4.4 se muestran los resultados considerando que el operador pronuncia una palabra al escuchar el sonido de alerta:

COMANDO	ACIERTOS	DESACIERTOS
Conectar	4	0
Fin	3	1
Adelante	4	0
Atrás	4	0
Paro	4	0
Giro Uno	2	2
Giro Dos	4	0
SUMA	**25**	**3**
	TOTAL CASOS	**28**

Tabla. 4.4 Resultados obtenidos para las pruebas finales

Los resultados de la tabla anterior se muestran en la figura 4.5.

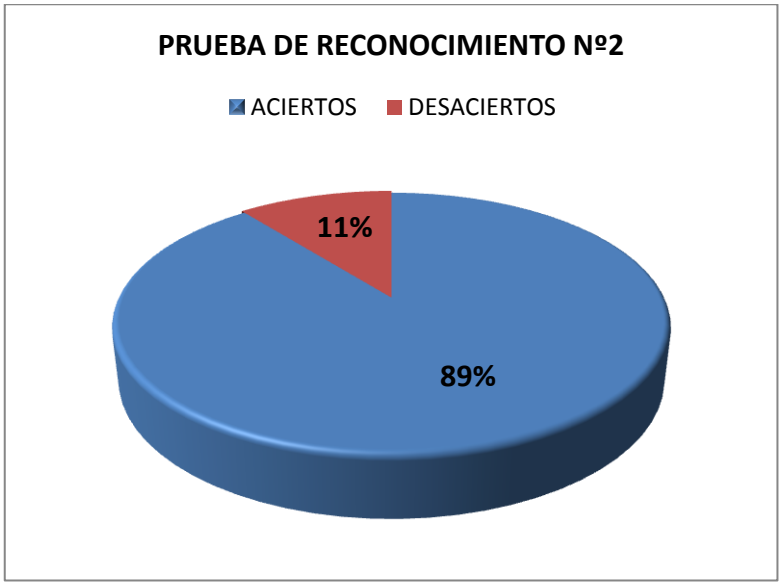


Tabla. 4.5 Taza de aciertos para pruebas finales

$$tasa\ de\ aciertos(\%) = \frac{25}{28} x100$$

$$tasa\ de\ aciertos(\%) = 89.29\%$$

Como se observa se ha mejorado la tasa de aciertos del sistema de reconocimiento un 21.44%. Ahora la tasa de aciertos es de 89.29%, lo cual es un valor aceptable.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

- El sistema de reconocimiento de voz requirió la implementación de una interfaz entre MATLAB y ARIA. Esta comunicación se implementó en código C++ mediante el uso de archivos MEX, que toman el dato de la palabra reconocida en MATLAB y la traducen a un comando directo de ARIA.

- Para realizar el reconocimiento es necesario analizar la señal de voz de manera segmentada para entender la evolución temporal, ya que si se usa una ventana muy grande los cambios locales se omiten y si es muy pequeña se reflejan demasiados cambios puntuales.

- Para analizar las características de una señal de voz es necesario únicamente extraer los parámetros que dependen del tracto vocal y excluir las señales de excitación de la voz.

- Para una toma mejor de la señal de voz se requiere el uso de un micrófono tipo diadema ya que mantiene un distancia fija con el interlocutor y esto reduce el error de salida del algoritmo de reconocimiento.

- El uso de MATLAB permitió reducir la complejidad del procesamiento digital de señales y ayudó a filtrar la voz para enfatizar únicamente las características acústicas, llamadas formantes.

- La interfaz gráfica desarrollada empleando la herramienta GUIDE de MATLAB tiene un diseño flexible, que puede ser mejorado a fin de incorporar nuevas funciones al sistema.

- La interfaz de comunicación fue escrita en lenguaje C++ con el compilador de Visual Studio .NET 2003 para evitar conflictos debido a que de esa manera están generadas las librerías de ARIA.

REFERENCIA BIBLIOGRÁFICA

- Borgstrom, J. (3 de Abril de 2005). *ARIA and Matlab Integration With Applications.* Recuperado el 21 de Noviembre de 2012, de http://www8.cs.umu.se/education/examina/Rapporter/JonasBorgstrom.pdf

- García, J. (12 de Noviembre de 2010). *Sistema de Reconocimiento de Palabras para un robot instrumentista*. Recuperado el 2013 de Febrero de 21, de http://electronicatecnoparque.blogspot.com/2010/11/sistema-de-reconocimiento-de-palabras.html

- Getreuer, P. (15 de Abril de 2010). *Writing MATLAB C/MEX Code.* Recuperado el 22 de Marzo de 2013, de http://classes.soe.ucsc.edu/ee264/Fall11/cmex.pdf

- Gómez, J. (1 de Octubre de 2011). *Procesamiento Digital de Señales de Voz.* Recuperado el 8 de Abril de 2013, de Modelos de Producción de Voz: http://www.fceia.unr.edu.ar/prodivoz/modelo_prod_voz.pdf

- Gutiérrez, D. (4 de Julio de 2012). *El Aparato Fonador*. Recuperado el 9 de Marzo de 2013, de http://es.scribd.com/doc/12848398/Funcionamiento-Del-Aparato-Fonador

- Kinnunen, T. H. (2011). *Speech Technology Workshop.* Autumn.

- MathWorks. (2013). *Supported and Compatible Compilers – Release 2009b.* Recuperado el 2 de Marzo de 2013, de http://www.mathworks.com/support/compilers/R2009b/

- MobileRobots. (Enero de 2006). *Pioneer 3 Operation Manual*. Recuperado el 9 de Febrero de 2013, de http://www.ist.tugraz.at/_attach/Publish/Kmr06/pioneer-robot.pdf

- Palacios, M. (24 de Enero de 2010). *Control de movimiento de un robot por medio de reconocimiento de voz.* Recuperado el 20 de Abril de 2013, de http://es.scribd.com/doc/60874392/Paper

- Posada, L. (27 de Abril de 2009). *MatlabAriaInterface*. Recuperado el 2012 de Diciembre de 2013, de http://www.rst.e-technik.tu-dortmund.de/cms/de/Forschung/Software/index.html

- Velásquez, G. (26 de Abril de 2008). *Sistema de reconocimiento de voz en MATLAB.* Recuperado el 14 de Marzo de 2013, de http://biblioteca.usac.edu.gt/tesis/08/08_0223_EO.pdf