UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

COMUNICACIÓN ENTRE UN DISPOSITIVO MOVIL Y UNA PC USANDO COMANDOS VERBALES

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN TELECOMUNICACIONES

PRESENTAN:

EDUARDO DANIEL CASTAÑENA TRUJILLO

FABIAN LOVERA GRANADOS

LIZBETH ANDREA RODRÍGUEZ ROMAN

DIRECTOR DE TESIS:

DR. JOSÉ ABEL HERRERA CAMACHO

MÉXICO, D.F *2009*

La escuela trata de prepararlo para el examen, pero la calificación final la da la vida

\- Dicho Popular -

Adjudíquense cada una de estas palabras que he escrito para todos ustedes: mi familia que son mis amigos, mis amigos que son mi familia y por supuesto Dios (Mahoma, Jehová, Alá, leyes de la Física o como usted quiera llamarle).

Gracias por crecer conmigo, por esas madrugadas y noches interminables, por acompañarme a los bailables y por ser como son. Gracias por las cajitas y los abrazos antes de los exámenes, por cuidarme y quererme tanto como yo a ustedes.

Gracias a mi madre por permitirme ser la razón de su vida, gracias a mi abue por dejar que la ame más allá del tiempo y la distancia, gracias a mi tío por cuidarme como solo él sabe hacerlo. Gracias a mis tíos (David y María de Jesús) por permitirse ser parte de los 7 Ramírez y siempre estar ahí, a mis primos (Jorge, Jeannete y David) por ser los hermanos que nunca tuve y por escucharme, de verdad los quiero mucho chicos. A mis padrinos Joaquín y Gloria por estar siempre al pendiente de mi a pesar de la distancia.

Gracias a ese cuadro de científicos donde mire mis sueños cada noche, a teatro, a la música de Cerati y mis locuras de fan. A mis compañeros de ruta: Adriana Peniche, Héctor Reyes, Bi, Pamela López, Ixchel Rivera, Ricardo Juárez, Dulce Chávez, Héctor Madariaga, Édgar Cruz y Angélica Ortega, en fin todo telecom y por supuesto a Fabián Lovera y a Daniel Castañeda y algún nombre que no mencionaré. Adry sin vos yo no sería la misma, gracias comadre por llorar y reír conmigo, por escucharme siempre y por ser mi mejor amiga, mi hermana. Fab, te quiero mucho, eres un tipo muy especial de verdad espero que realices todos y cada uno de tus sueños, Dany eres mi primo, mi mejor amigo y mi compañero de equipo, el que me lee la cara, sabes que siempre estaré para ti.

A la SATELFI, por dejarme una experiencia de vida, un sueño y amigos de verdad. A todo el departamento de Telecomunicaciones, por ser mis profesores, por apoyar a SATELFI, por llevar con orgullo la insignia de la Universidad y muy en especial al Mtro. Federico Sandoval, Dr. Miguel Moctezuma, Dr. Francisco García Ugalde, Mtro. Mario Ibarra y al Dr. Oleksandr Martiniuk que me dejaron mucho como seres humanos y profesores. Al H. Consejo Técnico de la Facultad de Ingeniería por permitirme apoyar a mis compañeros y enseñarme a amar aún más a la UNAM; sobre todo gracias al Mtro. Gonzalo Guerrero Zepeda y al Mtro. Víctor Pinilla. Al Dr. Abel Herrera por aceptar ser nuestro asesor de tesis y por apoyarnos en el Congreso IEEE, a todo el laboratorio de voz por compartir sus experiencias con nosotros y apoyarnos con nuestras dudas.

También un especial agradecimiento a Satmex por darme la primera oportunidad en el mundo de allá afuera, gracias al Ing. Pier Beaujean y al Mtro. Daniel Seseña por todo lo que compartieron conmigo y por sus enseñanzas.

Al espacio y al tiempo por permitirme ser una mujer ingeniería en una carrera estereotipada para hombres para poder romper con tabúes y aprender mucho de ellos.

Si tu nombre está o no carece de importancia, todos los nombres y cada uno de ellos están flotantes en mi mente y en mi agradecimiento. Los nuevos, los que siguen hasta hoy, los que vendrán, los de apodo, los que tienen mis apellidos, los que comen en mi casa, los que maúllan en mi casa, los que me cuidaron, los que espero, él que me espera, los que me dieron trabajo, los que me aguantan, los que se gastan el tiempo conmigo, los que cantan conmigo, los que me enseñaron, los que nunca fallan, los que me abrazan cuando

me ven, los que me cuidan desde el vientre, los que me quieren como soy, los que me hacen sonreír, los que hacen teléfonos, los que aman las telecomunicaciones,…..,todos, absolutamente todos los que han llegado a mi vida y los que se han ido de ella.

La propuesta de la Universidad me ha llevado a transformar lo que antes creía que no podía cambiar. En estos 5 años he compartido, con mis profesores y con mi familia, mis ignorancias, miedos y frustraciones. Pero también, hoy puedo decir al mundo que empiezo a superarlas.

Fab y Daniel, nosotros saldremos de aquí con una sola cosa que nadie más tiene, pero allí afuera habrá centenares de personas con el mismo título que nosotros; habrá miles haciendo lo que nosotros quisiéramos hacer para ganarnos el sostén. Pero recuerden que nosotros seremos los únicos que tendremos la custodia de nuestra propia vida.

Este es el principio en realidad, la espera termino y hoy se vislumbran los frutos de veinte años de estudio. Espero nunca defraudarlos, ser una gran profesionista y ser humano, un buen ingeniero para mi país y su apoyo para toda la vida.

Se me acaban las palabras y el espacio, pero sabes má que eres lo más importante en mi vida y abue sabes que vives este día conmigo como tú querías hacerlo.

GRACIAS TOTALES

Liz

Dedicada a todas esas luces que de día y de noche me muestran por donde es el camino. A Rafael y Paula, los mejores padres del mundo, esto es para ustedes. A Bere por contagiarme esos ánimos de nunca parar. A Sofía por todo su apoyo, inspiración y amor, gracias por estar ahí siempre bonita.

A Fabián y a Liz, gracias por compartir este paso tan importante, por sus ganas de crear, de sobresalir y de ser excelentes ingenieros. Ustedes hicieron esto posible, gracias por su enorme trabajo, dedicación y enorme talento. Mis mejores amigos.

A los maestros memorables de mi universidad. En especial al Dr. Abel por ser un gran profesor, pero sobre todo por creer en este proyecto y en nosotros, gracias por su apoyo, este trabajo es suyo profesor. Y a muchos compañeros y amigos, gente brillante e intachable, ustedes también forman parte de esto.

Daniel

Gracias a mis padres por educarme y apoyarme en mis acciones, gracias a los amigos que se convierten en hermanos como Liz, Lalo, Memo, Yuri, Dalia, Raúl, Alaide, SK, Julio, Mode, Che, Jazmin, Jessik, Israel, Rocío, Adrián, Ixchel, Toño, etc. Gracias por los momentos que hemos compartido y por los que faltan, gracias a los tíos, a mis maestros que se convierten en padres con sus buenos consejos, en verdad gracias a todos.
Y que todos nuestros sueños se hagan realidad.

Fab

Contenido

INTRODUCCIÓN

Las nuevas necesidades en los mercados han dado como resultado el incremento en el uso y desarrollo de tecnologías de comunicación cada vez más complejas, más pequeñas y con una mayor capacidad para realizar diferentes tareas con un solo dispositivo.

De acuerdo a la Unión Internacional de Telecomunicaciones (UIT), el teléfono móvil se ha convertido en el instrumento más popular dentro de estas tecnologías de la información, con cerca de 4,000 millones de usuarios en todo el mundo, frente a 1,200 millones de suscriptores a servicios de telefonía fija. En México al menos la mitad de la población cuenta con un teléfono celular.

El desarrollo de aplicaciones para dispositivos móviles y su reciente interacción con aplicaciones Web, ha permitido la creación de servicios que generan nuevos mercados, nuevas necesidades y nuevas tecnologías. El desarrollo de aplicaciones basadas en lenguajes multiplataforma, nos permite llevar software que opera exactamente igual en una Palm, un teléfono móvil o algún otro dispositivo portable.

Por otro lado, la interacción entre humanos se lleva a cabo de manera preferencial, bajo un modelo de comunicación hablado, por lo que se busca establecer esta característica en las comunicaciones humano-máquina.

Actualmente las computadoras carecen de características humanas perfeccionadas que les permitan hablar, escuchar, entender y aprender, y dado que el proceso hablar – escuchar es la base de las otras modalidades de desarrollo, es natural que sea una de las áreas con más avance e innovación científica y tecnológica. El potencial de los sistemas que funcionan bajo reconocimiento de voz es cada día más común, se pueden encontrar en dispositivos domésticos, sistemas operativos, aplicaciones Web e incluso en automóviles.

El objetivo del proyecto aquí presentado es el de crear una sistema de comunicación humano-máquina que utilice una interfaz móvil-servidor con un sistema reconocedor de voz para realizar procesos en el servidor que pueden servir para controlar dispositivos externos que interactúan por medio de internet.

Debido a que los dispositivos móviles cubren cada vez más necesidades y a que la expansión de Internet en cada proceso productivo toma mayor importancia, este sistema fue desarrollado para unir y explotar estos dos hechos. Nos valemos de la comunicación humano-máquina de una forma más dinámica y al mismo tiempo, complementamos los otros sistemas de comunicación como lo son los comandos escritos o instrucciones. Estas tecnologías no se excluyen sino que se complementan para hacer sistemas más sofisticados, funcionales y perfeccionar a los sistemas inteligentes.

El desarrollo de la tesis se generó de manera secuencial, la forma en la que se presentan los capítulos ejemplifica la manera en que se ha llevado a cabo este proyecto.

Capítulo 1. Reconocimiento de voz. En este capítulo se tratan bases del procesamiento de la voz, el modelo del tracto vocal, las características de las señales de voz, su clasificación, se explica también en qué consiste el reconocimiento de voz, la forma en que se debe de manejar una señal de voz para ser entrenada y reconocida.

Capitulo 2. Programación en el Móvil. En este capítulo se da una semblanza del lenguaje de programación utilizado para crear la aplicación en el dispositivo móvil. Se presenta la forma en la que este lenguaje opera, los requerimientos de hardware y software y la estructura de programación que debe seguir el desarrollo de una aplicación para desarrollo.

Capitulo 3. MATLAB. Este capítulo se enfoca a mostrar las herramientas con las que se trabajo el prototipo desarrollado en este software científico, mostramos la manera de generar funciones e interfaces gráficas necesarias para gestionar el motor de reconocimiento y entrenamiento de voz.

Capitulo 4. Programación en el Servidor Apache y PHP. Este capítulo describimos el Servidor Apache, su funcionamiento, estructura y potencial. Se presenta el lenguaje multiplataforma PHP, esto incluye sus características generales y el manejo de información. También se explica que es MySQL y la manera en la PHP gestiona información con esta base de datos.

Capitulo 5. Diseño y Desarrollo del Sistema de Comunicación. En este capítulo se explican todos y cada uno de los procesos que intervienen en el sistema de comunicación, tantos los que se llevan a cabo en el dispositivo móvil como los que se ejecutan en el servidor.

Capitulo 6. Pruebas y Conclusiones. En este capítulo se muestran tablas que contienen información de la evaluación del sistema, buscando probar su efectividad y velocidad. También se presenta un análisis del sistema en cuanto a su rendimiento y se plantean oportunidades de mejora. Finalmente presentamos conclusiones sobre todo el proyecto.

La tesis cuenta con apéndices que buscan cubrir temas que no son el objetivo de la misma pero que sin embargo están presentes en el proyecto. Además se incluye un manual de usuario del sistema para que pueda ser utilizado eficazmente.

1. RECONOCIMIENTO DE VOZ

1.1 Caracterización de las señales de voz

En este capítulo se presenta la descripción de los algoritmos de procesamiento digital de señales que componen el reconocedor empleado en esta tesis.

Para efectuar el reconocimiento de palabras es necesario entrenar las palabras que se quiera reconocer en un proceso que llamamos entrenamiento. Solo después de este entrenamiento es posible hacer el reconocimiento. Estos dos procesos están basados en el modelo digital de voz.

1.1.1 Sistema generador de voz

La voz humana se produce voluntariamente por medio del aparato fonador. Éste está formado por los pulmones, como fuente de energía en la forma de un flujo de aire, la laringe, que contiene las cuerdas vocales, la cavidad faríngea, las cavidades oral y nasal y una serie de elementos articulatorios: los labios, los dientes, el paladar, y la lengua.

Las cuerdas vocales son, en realidad, dos membranas dentro de la laringe orientadas de adelante hacia atrás. Por adelante se unen en el cartílago. Por detrás, cada una está sujeta a uno de los dos cartílagos aritenoides, los cuales pueden separarse voluntariamente por medio de músculos. La abertura entre ambas cuerdas se denomina glotis.

Cuando las cuerdas vocales se encuentran separadas, la glotis adopta una forma triangular. El aire pasa libremente y prácticamente no se produce sonido. Es el caso de la respiración.

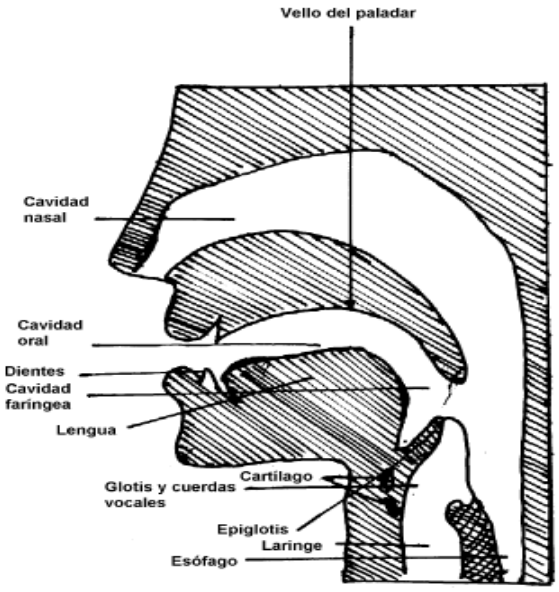


Figura 1.1 Aparato fonador

La emisión de sonidos se debe al funcionamiento de los siguientes elementos:

- Aire, el cual es expulsado por los pulmones
- Un vibrador sonoro constituido por las cuerdas vocales.
- Un resonador, formado por la boca, nariz y faringe.
- Articuladores, formado por labios, dientes, paladar y mandíbula.

En la laringe se produce la voz, tanto el tono fundamental y sus armónicos y se modifica en el resonador y en la boca se amplifica para finalmente formar el timbre. Los órganos articuladores moldean este sonido transformándose así en articulaciones (fonemas y silabas).

Después de que el aire que se expulsa de los pulmones, la laringe realiza una excitación que pueden ser fonaciones, susurros, fricaciones, compresiones o vibraciones:

- Fonación

La fonación es el resultado de la oscilación de las cuerdas vocales, cuando el aire pasa por ellas, estas vibran. Los sonidos resultantes de la fonación se llaman sonoros y los sonidos en ausencia de fonación se llaman sordos. Las vocales producen sonidos sonoros y letras como f, s, p y k son sonidos sordos.

- Susurros

Estos son generados en la laringe, las cuerdas vocales no cierran la glotis completamente y en esa abertura el aire que pasa genera turbulencias que generan sonidos de banda ancha.

- Fricación

Es similar al susurro en cuanto a su generación, pero existe un lugar de articulación adicional en el tracto vocal, generando sonidos fricativos. La fricación puede ser con o sin fonación.

-Compresión

Se produce cuando el tracto vocal está cerrado y el hablante sigue exhalando, la presión aumenta y se produce un transitorio. Si este transitorio es abrupto el sonido producido es una plosiva “p” y si es gradual se produce un sonido parecido al fricativo llamado africativo por ejemplo la “j”.

- Vibración

Las vibraciones ocurren en muchos lugares del tracto vocal, por ejemplo la letra “r” involucra la vibración de la lengua contra el paladar.

1.1.2 Modelo digital de la voz

Para representar la voz necesitamos modelarla matemáticamente para ello hacemos uso del modelo propuesto por Fant [1] y Flanagan [2], este modelo de la voz incluye: la señal de excitación, la respuesta del tracto vocal y los efectos de los labios.

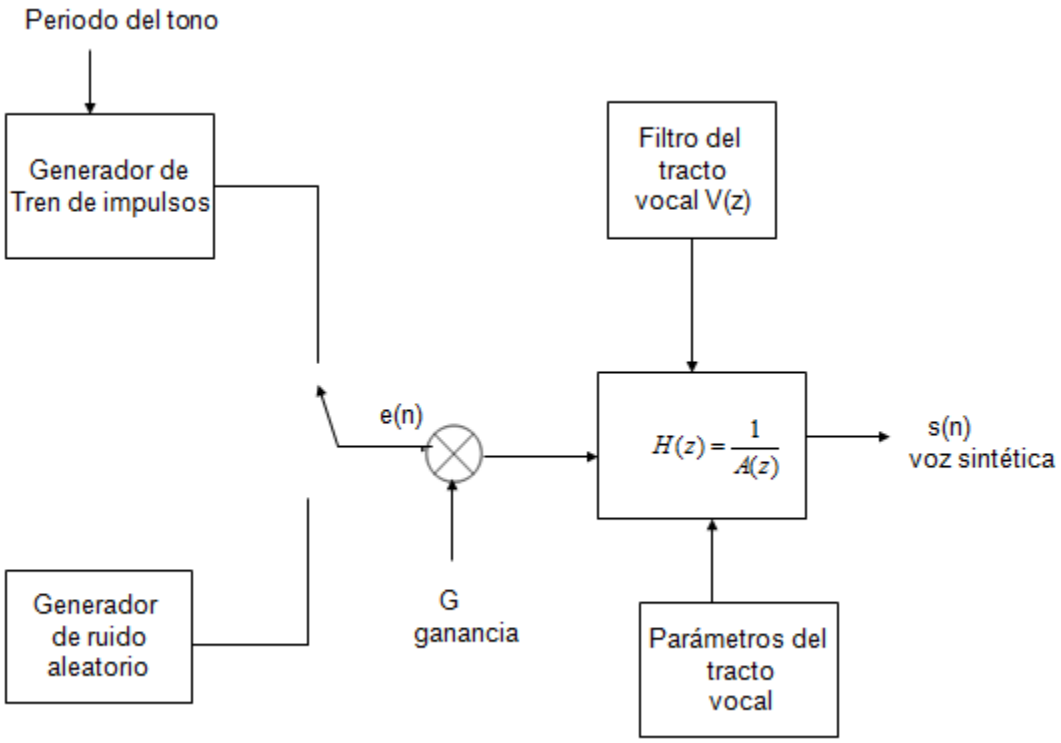


Figura 1.2 Modelo digital de la voz

Este modelo nos permite representar un sonido de voz en un conjunto de coeficientes principalmente. Este sistema se divide en dos, la primera parte es un sistema que es excitado por un tren de pulsos o bien por un generador de ruido aleatorio, la segunda parte es la modulación por el tracto vocal.

La excitación glotal es la entrada del filtro. Los cinco tipos de excitación, antes mencionados, son reducidos en dos señales periódicas (o sonidos sonoros) y ruido turbulento (o sonidos sordos). El espectro glotal es un tren de impulsos espaciados a frecuencias iguales a la frecuencia fundamental.

$$s(n)=\sum_{i=1}^{p} a_k s(n-k)+Gu(n) \qquad (1.1)$$

Y su función de transferencia es la siguiente:

$$H(z)=\frac{G}{1-\sum_{k=1}^{N}\alpha_k z^{-k}} \qquad (1.2)$$

Donde G es la ganancia total, α_i la ubicación de los polos, estos son las resonancias de la voz.

Para sonidos sonoros el sistema es excitado por un tren de pulsos y si deseamos sonidos sordos usamos ruido aleatorio. Se pueden representar satisfactoriamente casi todos los sonidos de la voz si se utiliza un filtro todo polos con un orden suficientemente alto.

- Fonemas

La clasificación de los fonemas en español se muestra en la figura 1.3:

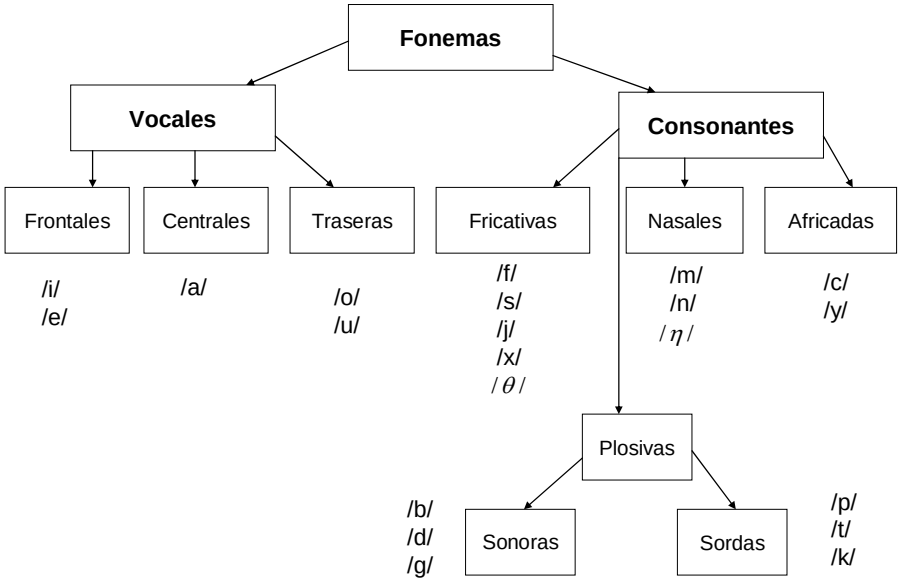


Figura 1.3 Clasificación de los símbolos fonéticos del español de México

Los fonemas vocálicos son las cinco vocales de nuestro alfabeto, en la tabla 1.1 podemos observar el rango de frecuencias en las que estas se encuentran, siendo f_o la frecuencia central.

Vocal	f_0	f_1	f_2
A	900	1300	2100
E	375	2200	2550
I	325	2300	3900
O	400	550	4300
U	325	425	4500

Tabla 1.1 Frecuencias de los fonemas vocálicos

Los diptongos son monosílabos que empiezan cerca de la posición articulatoria de una vocal seguida de otra vocal. La vocal con mayor apertura es el núcleo silábico y la otra es la silaba marginal. Hay diptongos crecientes y decrecientes. Los crecientes, a su margen silábico se les llama semiconsonantes, por ejemplo j y w. Mientras que los diptongos decrecientes a su margen silábico se le llama semivocal, los cuales son distintas de las semiconsonantes por su posición en el diptongo por ejemplo i y u.

Las consonantes se clasifican de acuerdo a la forma en que se articulan, esta clasificación la podemos observar en la tabla 1.2.

Clasificación	Fonema	Forma de articulación	Lugar de articulación	Concentración de energía (Espectro)	Ejemplo	Alófonos	Ejemplo
Plosivas	/b/	sonora	labial	500-1500 Hz (débil)	un vaso	[β]	el vaso
	/p/	sorda	bilabial		ópera		
	/g/	sonora	velar	1500-4000 Hz (concentrado)	un gusto	[ɣ]	ese gusto
	/k/	sorda	velar		casa		
	/d/	sonora	dental	4000- Hz (fuerte)	el diente	[ð]	ese diente
	/t/	sorda	dental		tela		
Fricativas	/j/	sonora	palatal	0-600, 2200-3000 Hz	mayo	[dz]	cónyuge
	/f/	sonora	labiodental	0-400, 1400-2200, 2900-4000 y 6000-8000 Hz	café		
	/θ/	sonora	interdental	0-500, 2600-3600 y 5000-8000 Hz	caza		
	/s/	sorda	alveolar		mesa	[z]	desde
	/x/	sorda	velar	0-900 Hz	caja		
Africativas	/ č /	sorda	palatal	altas frecuencias	chile		
Nasales	/m/	sonora	bilabial	altas frecuencias	mesa		
	/n/	sonora	velar	altas frecuencias	nariz	([m]) [M] [n] [ɱ] [n] [N]	enviar enfermo lanzar cuento ancho hongo
	/□/	sonora	bilabial	altas frecuencias	niño		
Semivocales Laterales	/ʎ/	sonora	palatal		llego		
	/l/	sonora	alveolar		lado	[l] [˙]	dulce el toro
Semivocales Vibrantes	/r/	sonora	alveolar		cara		
	/ r̄ /	sonora	alveolar		rezo		

Tabla 1.2 Clasificación de las consonantes

- **Africadas:** se producen con un cierre en el tracto vocal seguido de una expiración que produce turbulencia.
- **Aspiradas:** El tracto vocal está cerrado en la articulación y se exhala aire al siguiente sonido.
- **Fricativas:** el tracto vocal está abierto parcialmente en la articulación y el paladar cerrado y se genera ruido en la articulación.

- **Laterales:** El tracto vocal está cerrado en el punto de la articulación.
- **Nasales:** El tracto vocal está cerrado en el punto de la articulación y el paladar abierto.
- **Plosivas:** El tracto vocal está cerrado en el punto de la articulación, el pasaje nasal está cerrado y existe una exhalación constante.
- **Semivocales:** El tracto vocal esta semiabierto en el punto de la articulación.
- **Vibrato:** Hay una abertura y cierres oscilatorios en la articulación, seguidos de una exhalación constante.

1.2 Entrenamiento

En la etapa de entrenamiento el usuario graba cada palabra que desea reconocer varias veces, después se calcula un modelo, con esas palabras, para poder reconocerla por medio de técnicas de procesamiento.

En la figura 1.4 podemos ver el diagrama de bloques de entrenamiento de palabras aisladas.

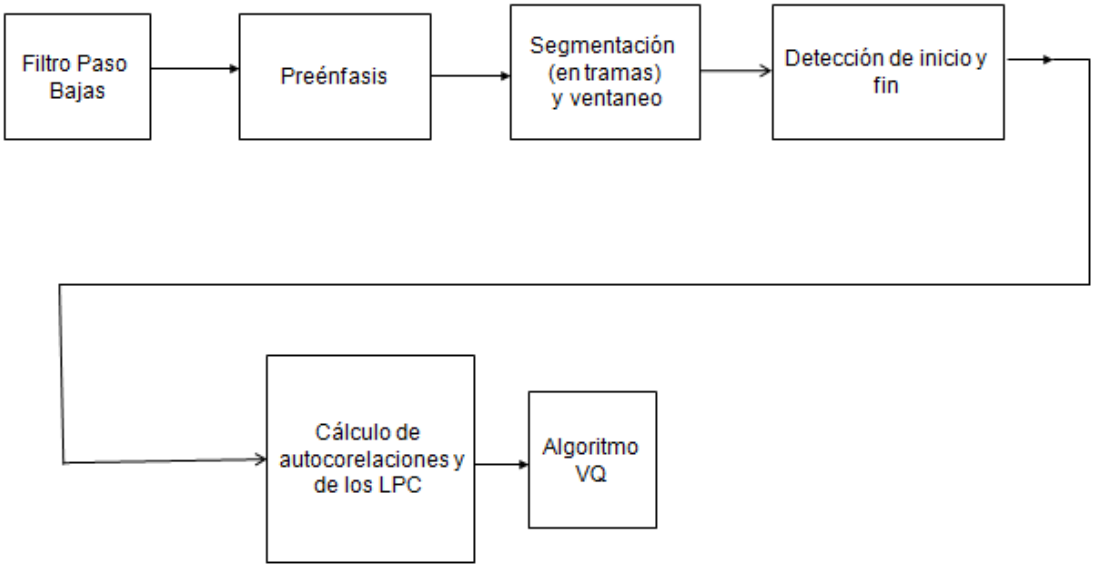


Figura 1.4 Diagrama de bloques del algoritmo de entrenamiento

1.2.1 Filtro Paso Bajas

Este filtro se encarga de limitar la señal en frecuencia y además atenúa las componentes de alta frecuencia del ruido.

1.2.2 Preénfasis

El preénfasis es un procesamiento de la señal que nos permite resaltar algunas de sus características las cuales pudieron ser atenuadas por un transductor (como el micrófono), pues

estos pueden tener una mejor respuesta en frecuencias bajas, por lo tanto, el filtro preénfasis nos permite aumentar las frecuencias altas donde se encuentra la mayor parte de la información auditiva, la cual se utiliza para reconocer la palabra.

El preénfasis se puede implementar con un filtro analógico paso altas cuya frecuencia de corte debe ser de 3 dB, en algún lugar entre 100 Hz y 1 KHz. Este filtro implementarse de manera digital por medio un filtro paso altas cuya ecuación en diferencias es la siguiente:

$$y[n] = x[n] - ax[n-1] \qquad (1.3)$$

Y su función de transferencia es la siguiente:

$$H(z) = 1 - az^{-1} \qquad 0.9 \le a \le 1 \qquad (1.4)$$

Y el diagrama de bloques que representa a este filtro es el siguiente:

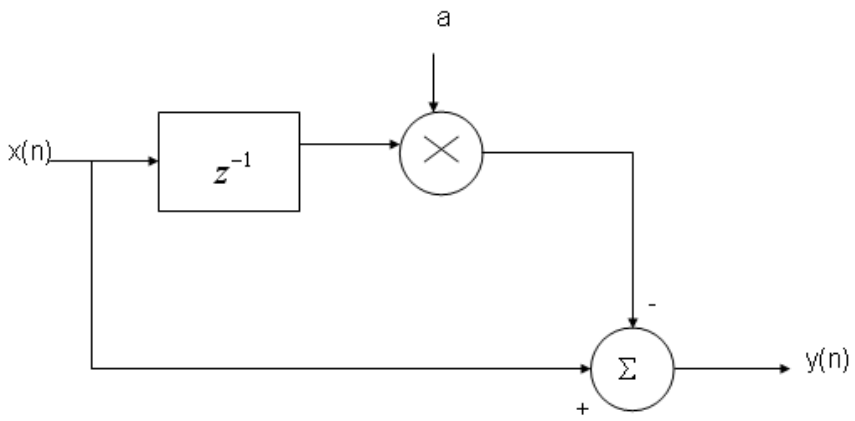


Figura 1.5 Filtro digital de preénfasis

1.2.3 Bloques de ventaneo

La función del método de Codificación de Predicción Lineal (LPC) es la de representar la envolvente espectral de una señal digital de voz utilizando un modelo de predicción lineal.

Sin embargo, este método solo es aplicable a señales estacionarias, por lo cual, es necesario dividir la señal en segmentos cuasi-estacionarios llamados ventanas antes de calcular el LPC. Cuando una señal de voz se analiza en periodos de tiempo muy cortos (de 5 a 100 ms), la variación de la señal con respecto al tiempo es muy pequeña y puede considerarse cuasi estacionaria, Deller [3].

Por lo anterior, la señal de voz se secciona en pequeños segmentos llamados tramas, esto se hace multiplicando al señal de voz s(m) con una ventana w(m) la cual tiene como característica que es

cero fuera del intervalo en que deseamos extraer la señal. Con el ventaneo se suavizan los espectros de cada trama.

Algunos ejemplos de ventanas son: rectangular, Bartlett(triangular), Hamming, Hanning y Blackman. Para la realización de esta tesis se utilizo la siguiente ventana:

$$\text{Hamming} \quad \omega[m]\begin{cases} 0.54-0.46\cos\left(\dfrac{2\pi m}{N}\right) & 0\le m\le N-1 \\ 0 & eoc \end{cases} \qquad (1.5)$$

En este caso N es el tamaño de la ventana en el tiempo.

1.2.4 Detección de inicio y fin de palabra

El objetivo del procesamiento de la señal es obtener una representación más útil de la información. Las técnicas de procesamiento llamadas Métodos en el dominio del tiempo son aquellas en las que se involucra la forma de onda de la señal, entre estos métodos encontramos: la tasa promedio de cruces por cero, energía y la función de autocorrelación.

La energía en tiempo corto de la señal de voz provee una representación que refleja estas variaciones en amplitud. La energía en tiempo corto se define como:

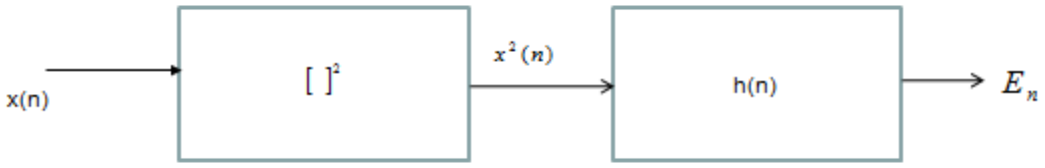
$$E_n = \sum_{m=-\infty}^{\infty}\left[x(m)w(n-m)\right] \qquad (1.6)$$

Que también puede ser escrita como:

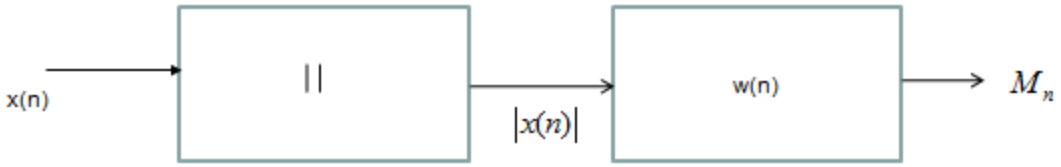
$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)h(n-m) \qquad (1.7)$$

donde:

$$h(n) = w^2(n) \qquad (1.8)$$



(a)



(b)

Figura 1.6 (a) Representación de la energía en tiempo corto y (b) Representación de la magnitud promedio en tiempo corto

Como se observa en la figura 1.6 (a), la señal $x^2(n)$, es filtrada por un filtro lineal con respuesta al impulso h(n). La elección de esta respuesta al impulso, o equivalentemente la ventana, determina la naturaleza de la representación de la energía en tiempo corto. Deseamos que dicha ventana sea de corta duración y que responda a los cambios rápidos de amplitud.

El significado de calcular E_n es que provee la base para distinguir entre segmentos de voz sonoros de segmentos de voz sordos. Los valores de E_n para segmentos sordos son mucho menores que los de segmentos sonoros. La función de energía también se usa para localizar el tiempo en el cual la voz sonora se vuelve sorda y para la voz de alta calidad (con alta relación señal a ruido), la energía puede ser usada para distinguir la voz del silencio.

Sin embargo, existe una dificultad con la función de la energía y es que es muy sensible a niveles altos de señal (debido a la elevación de la señal al cuadrado), por ello enfatiza las variaciones grandes entre muestra y muestra. Para solucionar esto se define la magnitud promedio:

$$M_n = \sum_{m=-\infty}^{\infty} |x(m)| w(n-m) \qquad (1.9)$$

Donde se calcula la suma pesada de los valores absolutos de la señal en lugar de la suma de los cuadrados. Para el cálculo de la magnitud promedio, el rango dinámico (relación entre el máximo y mínimo) es aproximadamente la raíz cuadrada del rango dinámico del cálculo de la energía estándar. Por ello, las diferencias en nivel entre regiones sonoras y sordas son menos pronunciadas como cuando se usa la energía en tiempo corto.

Para las señales discretas un cruce por cero ocurre cuando muestras sucesivas tienen signos algebraicos diferentes. La frecuencia de ocurrencia depende de la frecuencia de la señal F_0, muestreada a una frecuencia F_S, tiene $\frac{F_S}{F_0}$ muestras en cada ciclo. Cada ciclo tiene dos cruces por cero, la frecuencia promedio de cruces por cero se puede obtener como:

$$Z = 2\frac{F_S}{F_0}\left[\frac{cruces}{muestra}\right] \qquad (1.10)$$

Las señales de voz son de banda ancha y su interpretación por la tasa promedio de cruces por cero es menos precisa pero se pueden tener buenas estimaciones.

En la voz, las altas frecuencias implican altas tasas de cruces por cero y bajas frecuencias implican bajas tasas de cruces por cero, existe una alta correlación entre la tasa de cruces por cero y la distribución de la energía con la frecuencia. Se dice que si la tasa de cruces por cero es alta, la señal de voz es sorda, pero si la tasa de cruces por cero es baja la señal de voz es sonora.

La tasa promedio de cruces por cero media es del 49 por 10 ms para voz sorda y 14 por 10 ms para voz sonora, debido a que hay traslape no es posible tomar una decisión sobre la señal de voz, por ello se tiene que tomar unas consideraciones de este método.

La tasa de cruces por cero es afectada por el offset de DC en el convertidor analógico-digital y por el ruido, para minimizar estos efectos se usa un filtro paso banda en lugar de un paso bajas como filtro antialiasing para eliminar las componentes DC, también se puede considerar el periodo de muestreo pues este determina la resolución en el tiempo. Una resolución fina requiere una tasa de muestreo alta pero para el método de cruces por cero únicamente se requiere 1 bit en la cuantización.

Localizar el inicio y fin de un segmento de voz es de suma importancia en el procesamiento además de la discriminación de la señal con el ruido de fondo. A pesar de estas dificultades, la representación de la energía y de la tasa de cruces por cero combinadas forman un algoritmo para detectar el inicio y fin de una señal de voz.

La tasa de cruces por cero la podemos representar como:

$$Z_n = \sum_{m=-\infty}^{\infty} \left|\operatorname{sgn}\left[x(m)\right] - \operatorname{sgn}\left[x(m-1)\right]\right| w(n-m) \qquad (1.11)$$

Y la magnitud promedio como

$$M_n = \sum_{M=-\infty}^{\infty} \left|x(m)\right| w(n-m) \qquad (1.12)$$

En el algoritmo se obtienen esas dos expresiones en un intervalo de grabación a una tasa de 100 veces/s. Se asume que los primeros 100 ms del intervalo no contienen voz. La media y desviación estándar de la magnitud promedio y la tasa de cruces por cero se calculan para ese intervalo para caracterizar el ruido de fondo. Usando esta caracterización estadística y la magnitud promedio máxima del intervalo, se calculan los umbrales de energía y de cruces por cero. Los puntos de inicio y fin se encuentran fuera de este intervalo.

1.2.5 Autocorrelación

La autocorrelación se define como la correlación de la señal consigo misma y es de gran utilidad para encontrar patrones repetitivos dentro de una señal, por ejemplo, la periodicidad de una señal con ruido. Después de ventanear la señal se autocorrelaciona del siguiente modo:

$$r_1(m) = \sum_{N-1-m}^{n=0} \tilde{x}_1(n) \cdot \tilde{x}_1(n+m) \qquad m = 0,1,....., p, \qquad (1.13)$$

Donde p es el orden de los coeficientes LPC.

1.2.6 Análisis de predicción lineal

El método de predicción lineal es de las mejores técnicas en el análisis de voz ya que nos permite estimar parámetros como la entonación, los fonantes, el espectro y así determinar si se trata de sonidos sonoros o sordos. En los primeros se abren y cierran las cuerdas vocales , modificando el área de la traquea y produciendo un tren de impulsos casi periódicos; para los segundos se representa como el aire que fluye libremente hasta alcanzar el tracto vocal cuando permanecen abiertas las cuerdas vocales y representa una estructura ruidosa produciendo así ruido aleatorio.

La idea de este método es que una muestra de la señal se debe aproximar a la combinación lineal de las muestras anteriores, esto se logra minimizando la sumatoria del error cuadrático medio entre las muestras de voz real y las señales de voz predecidas, así se logra determinar el conjunto de coeficientes predictores, estos coeficientes son los pesos que se les da a los coeficientes usados en la combinación lineal que forma la predicción.

Un predictor de la señal de salida en el instante *n* puede calcularse como:

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \qquad (1.14)$$

donde a_k son los coeficientes predictores.

Por lo tanto, la función de un sistema de un predictor lineal cuyo orden es p, se puede expresar como:

$$P(z) = \sum_{k=1}^{p} a_k z^{-k} \qquad (1.15)$$

Se define el error de predicción $e(n)$ como:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k \cdot s(s-k) \qquad (1.16)$$

La ecuación anterior tiene como función de transferencia:

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \qquad (1.17)$$

Observando la ecuación de error de predicción sabemos que obedece al modelo de voz de la ecuación 1.1., por lo tanto e(n)=Gu(n). Así el error del filtro de predicción A(z) será un filtro inverso para el sistema H(z) de la ecuación (1.1). En LPC el tracto vocal es modelado como un filtro digital todo polos, donde p es el orden del filtro.

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a_1 z + \ldots + a_p z^{-p}} \qquad (1.18)$$

El problema consiste en encontrar los coeficientes predictores en una señal de voz que varia en el tiempo, los coeficientes predictores deben ser estimados en segmentos cortos de voz. Para atacar este problema se debe reducir el error cuadrático sobre ese segmento corto de voz. Los coeficientes resultantes se asumirán como los parámetros de una función H(z) en el modelo de producción de voz.

El error cuadrático medio en un segmento corto se define como E_n que se obtiene de la sumatoria considerando la suma de los errores y la ecuación de ventaneo aplicada a la señal. Considerando que $s_n(m)$ es diferente a cero para $0 \le m \le M-1$, entonces el error de predicción $e_n(m)$ para n predictor de orden p puede ser distinto a cero en el intervalo $0 \le m \le N-1+p$. Por lo tanto, E_n se expresa de la siguiente manera:

$$E_n = \sum_{m=0}^{N+p-1} e_n^2(m) \qquad (1.19)$$

$$E_n = \sum_{m=0}^{N+p-1} \left[s(n) - \sum_{k=1}^{p} a_k s(n-k) \right]^2 \qquad (1.20)$$

Se define $s_n(n)$ como un segmento de voz que es seleccionado en la vecindad de la muestra n como:

$$s_n(n) = s(m+n) \qquad (1.21)$$

Para resolver la ecuación anterior es necesario deriva parcialmente E_n con respecto a cada a_k y el resultado se iguala a cero.

$$\frac{\partial E_n}{\partial a_k} = 0 \qquad k = 1, 2, \ldots p \qquad (1.22)$$

Resultando:

$$\sum_{m=0}^{N+p-1} s_n(m-i) \cdot s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_{m} s_n(m-i) \cdot s_n(m-k) \qquad 1 \le i \le p \qquad (1.23)$$

Donde $\hat{a_k}$ representa a los valores de a_k que minimizan E_n. Se define como:

$$\phi_n(i,k) = \sum_{m=0}^{N+p-1} s_n(m-i)s_n(m-k) \qquad (1.24)$$

De ese modo, podemos reescribir la ecuación (1.23) de la siguiente manera:

$$\sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(i,0) \qquad i = 1,2,...,p, \qquad (1.25)$$

El conjunto de esa p ecuaciones con sus p incógnitas se pueden resolver de diversas formas para minimizar el promedio cuadrático del error de predicción. Usando las ecuaciones (1.20) y (1.23), expresamos el error cuadrático medio como sigue:

$$E_n = \sum_{m=0}^{N+p-1} s_n^2(m) - \sum_{k=1}^{p} a_k \sum_{m=0}^{N+p-1} s_n(m)s_n(m-k) \qquad (1.26)$$

Con ayuda de la ecuación (1.22) , E_n se puede expresar como:

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} a_k \phi_n(0,k) \qquad (1.27)$$

Como se observa en la ecuación superior, el error mínimo depende de un componente fijo y de un componente que depende de los coeficientes predictores. Para encontrar los coeficientes es necesario calcular las cantidades $\phi_n(i,k)$ para $0 \le k \le p$, posteriormente se encuentran los a_k resolviendo la ecuación (1.26). El cálculo es algo complicado, es necesario utilizar varios procedimientos numéricos y uno de ellos es el algoritmo de Levinson-Durbin.

Análisis LPC

Para convertir los coeficientes de autocorrelación en coeficientes LPC es necesario usar el método de Levinson-Durbin, el cual se puede expresar como sigue:

$$E^{(0)} = r(0) \qquad (1.28)$$

$$k_i = \frac{\left\{ r(i) - \sum_{j=1}^{L-1} \alpha_j^{(i-1)} r\left(|i-j|\right) \right\}}{E^{(i-1)}} \quad , \quad 1 \le i \le p \qquad (1.29)$$

$$\alpha_i^{(i)} = k_i \qquad (1.30)$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \quad , \quad 1 \le j \le i-1 \qquad (1.31)$$

$$E^i = (1-k_i^2)E^{(i-1)} \qquad (1.32)$$

Se resuelve así el conjunto de p ecuaciones y la solución son los coeficientes LPC.

1.2.7 Cuantización vectorial (VQ)

La representación LPC de una señal nos da un conjunto de vectores que contienen las características de su espectro variando en el tiempo. Para poder una señal de voz es necesario comparar una serie de vectores de entrenamiento contra una serie de vectores a reconocer y además cuantificar la distorsión entre ellos. Esto es algo impractico por lo que es necesario utilizar la cuantización vectorial (VQ).

VQ trata con señales de entrada digitales y es usada con el propósito de comprimir la información, también puede ser vista como una forma de reconocimiento de patrones donde un patrón de entrada es "aproximado" mediante un elemento de un conjunto de patrones estándares predeterminados.

Un cuantizador vectorial Q de dimensión k y tamaño N mapea un vector en un espacio euclidiano k-dimensional, R^k, en un conjunto finito C el cual contiene N puntos de salida llamados centroides. Por lo tanto:

$$Q : R^k \rightarrow C \qquad (1.33)$$

donde $C = \{y_1, y_{2,...,} y_n\}$ y $y_i \in R^k$ para cada $i \in \square = \{1, 2, ..., N\}$. El conjunto C es el alfabeto o code book y es de tamaño N, lo cual significa que posee N elementos distintos, cada uno siendo un vector R^k.

VQ además de comprimir la información, crea un vector promedio el cual permite que el usuario no repita una palabra exactamente igual a la que se entreno ya que el vector cuantizado contiene los vectores de varios entrenamientos y con ello se disminuye el tiempo de procesamiento.

Para realizar una cuantización vectorial es necesario:

- Un gran conjunto de vectores de análisis espectral
- Una distancia, de un par de vectores
- Un algoritmo para encontrar un centroide
- Un algoritmo de clasificación para vectores arbitrarios que clasifique el vector del code book que sea más cercano al vector de entrada y que use el índice del code book como la representación espectral resultante.

El conjunto de vectores se obtiene a partir del entrenamiento de palabras. La calidad del entrenamiento dependerá de como y quien habla, en que condiciones y los micrófonos a utilizar.

Distancia de Itakura - Saito

Se trata de una medida de distancia cuando los sonidos se suponen reducidos por un sistema todo polos que responde a la siguiente ecuación en diferencias:

$$x[n]+a_1x[n-1]+...+a_p[n-p]=e[n] \qquad (1.34)$$

Donde $e[n]$ es una señal pseudo-periódica.

La energía residual resultado de filtrar una trama de voz con el inverso del filtro todo polos puede calcularse como:

$$\alpha(a)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\sum_{i=0}^{p}a_i e^{-jwi}\right|^2 S(w)dw=a^tR_pa \qquad (1.35)$$

La distancia de Itakura-Saito se define como:

$$d_{IS}(S,S')=\log\left(\frac{a^tR_pa}{a^tR_pa_p}\right) \qquad (1.36)$$

Algoritmo K-medias

El método de análisis de conglomerados de K medias es un método de agrupación de casos basado en distancias entre las variables. Se inicia seleccionando K casos, los más distantes entre sí, y se inicia la lectura del archivo de datos y se asigna como centro el más próximo y se va actualizando el valor de estos centros conforme se incorporan nuevos casos, ya que todos los casos se asignan a los conglomerados o grupos correspondientes se calculan los centroides de cada grupo.

Este algoritmo es fácil de implementar y es eficiente pues converge en pocas iteraciones. Entre sus defectos esta la dependencia de los valores iniciales los cuales son aleatorios.

La función de criterio es la siguiente:

$$J=\sum_{i=1}^{K}\sum_{n=1}^{N}M_{in}\left\|X(n)-C_i\right\| \qquad (1.37)$$

Donde N es el número de centroides, $\|..\|$ es la distancia euclídea, x(n) es el centroide de entrada n y M_{in} es la función de pertenencia que vale 1 si el centroide es el más cercano a ese grupo y cero en cualquier otro caso.

1. Inicialización: Se seleccionan arbitrariamente M vectores como el conjunto inicial de palabras código del code book.

2. Búsqueda del vecino más cercano: Para cada vector, se encuentra el vector más cercano y se asigna este vector a la célula correspondiente.

3. Actualización del centroide: Se actualiza el vector del código usando el centroide de los vectores de entrenamiento a esta célula.

4. Iteración: Se repiten los pasos 2 y 3 hasta que se llegue a una distancia umbral.

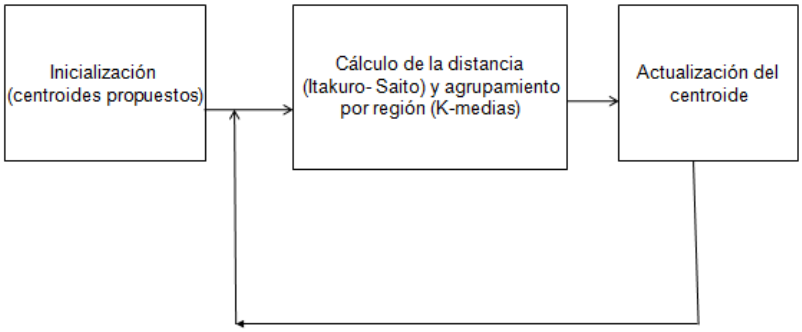


Figura 1.7 Algoritmo VQ

El comportamiento de este algoritmo depende de:

- La elección de los centroides iniciales
- El orden en que se toman las muestras
- Las propiedades geométricas de la información

1.3 Reconocimiento

Para poder reconocer una palabra se siguen los siguientes pasos:

1. Se pasa la señal de voz a reconocer a través del filtro de preénfasis, del ventaneo y se detecta su inicio y fin, con ello obtenemos una señal con espectro aplanado y estacionaria.

2. Por medio del algoritmo de Itakura-Saito se calcula la distancia de cada vector de la palabra a todos los vectores de cada code book.

3. Se toma la distancia mínima de cada vector de la palabra a reconocer a cada vector de los code books.

4. Se suman las distancias mínimas que se obtuvieron en cada code book.

5. Se selecciona la code book que representa a la palabra la que tiene menor distancia sumada.

2. PROGRAMACIÓN EN EL MÓVIL

2.1 Introducción a Java 2 Micro Edition

Java 2 Micro Edition (J2ME) es la versión del lenguaje Java orientado al desarrollo de aplicaciones destinadas a dispositivos con capacidades restringidas tanto en la capacidad de memoria disponible como en la capacidad de procesamiento. Estas son características típicas de dispositivos como teléfonos celulares, PDAs, dispositivos electrónicos inteligentes, dispositivos de navegación de coches, etc.

En la figura 2.1 muestra un esquema de los distintos dispositivos soportados por las diferentes ediciones de la plataforma Java 2, ilustrando el lugar que ocupa J2ME en el conjunto de la plataforma.

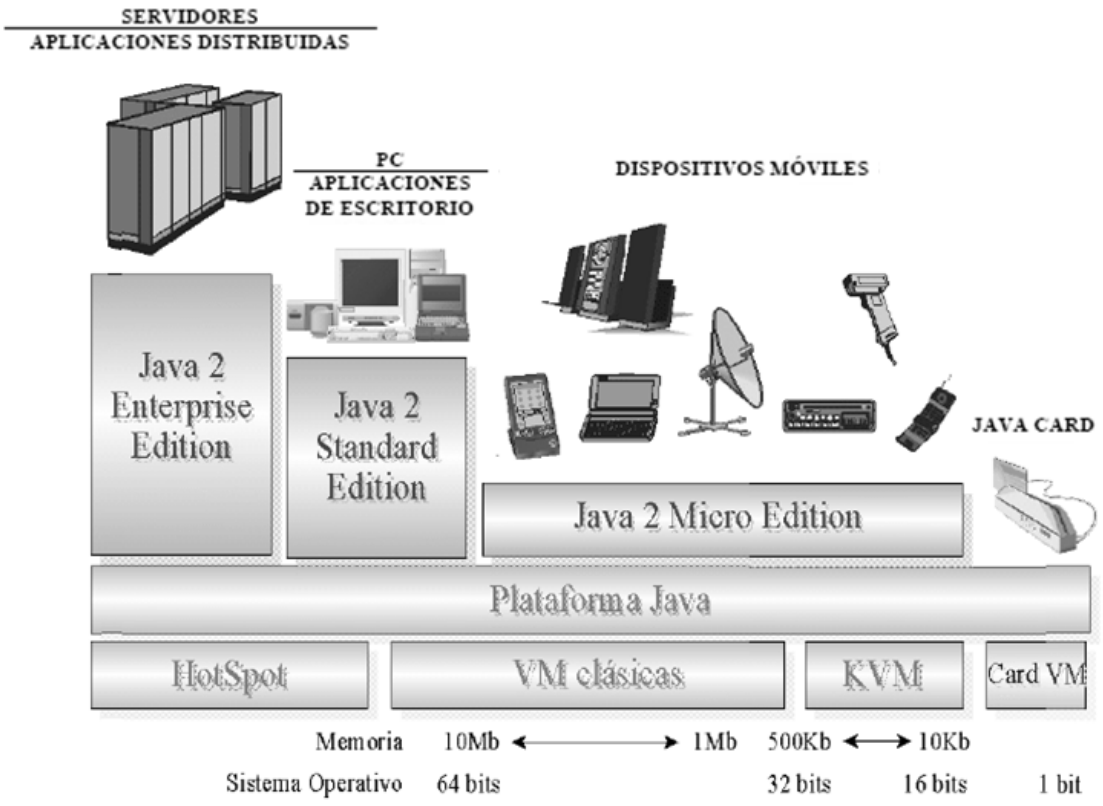


Figura 2.1 Arquitectura de la plataforma Java 2 de Sun Microsystems

En la figura 2.2 podemos ver que J2SE (Java 2 Standard Edition) es un subconjunto de J2EE (Java 2 Enterprise Edition) y J2ME es un subconjunto de J2SE excepto por el paquete ***javax.microedition.****. Es decir, las distintas versiones de la plataforma Java comparten paquetes en común.

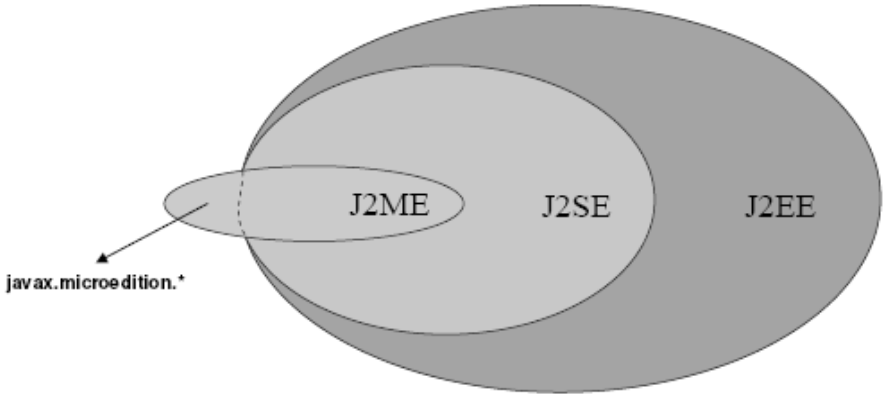


Figura 2.2 Estructura de Java 2 de Sun.

2.2 Arquitectura de J2ME

Ya hemos visto qué es Java Micro Edition y la hemos enmarcado dentro de la plataforma Java 2. En este apartado vamos a ver cuales son los componentes que forman parte de esta tecnología.

La arquitectura J2ME esta diseñada con especial hincapié en la escalabilidad y flexibilidad, porque no es posible predecir hoy qué dispositivos pueden crearse en el futuro y sin embargo, la arquitectura J2ME deberá estar lista para adaptarse a ellos. La arquitectura J2ME es, como consecuencia de esta filosofía, modular y escalable; características que están definidas en un entorno global constituido por cuatro capas básicas montadas sobre el sistema operativo del dispositivo, estas capas las podemos observar en la figura 2.3.

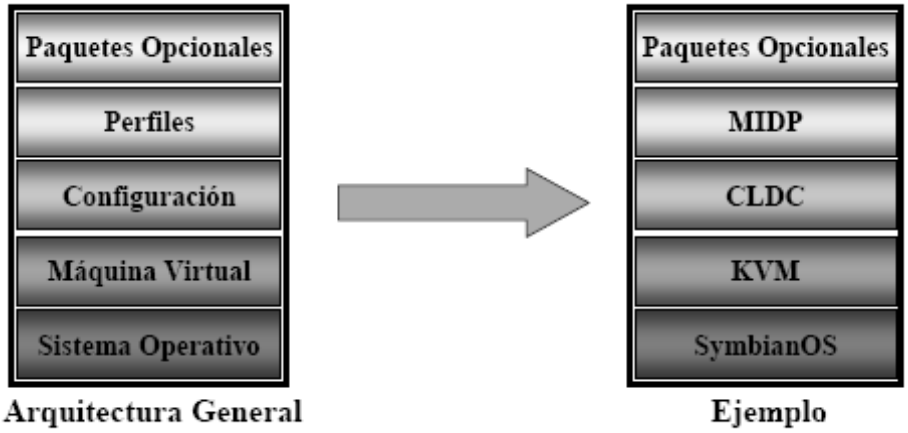


Figura 2.3 Arquitectura de J2ME

La capa correspondiente a la maquina virtual de Java corresponde a una implementación específica de la máquina virtual para cada uno de los dispositivos, de forma que puedan adaptarse y soportar el sistema operativo que se ejecute en este dispositivo.

- La capa de configuración esta orientada al dispositivo, define el mínimo conjunto de características de la maquina virtual Java y de las librerías y clases Java que están disponibles para un conjunto de dispositivos. Es la capa que define la parte común de las

librerías y características Java que van a estar disponibles para un conjunto especifico de dispositivos.

- La capa de perfil está orientada a la aplicación, define el mínimo conjunto de APIs disponible para una determinada familia de dispositivos. Las aplicaciones se escriben para un perfil en específico, de modo que cualquier dispositivo que soporte este perfil pueda ejecutarla.
- La capa de paquetes opcionales es específica de cada uno de los fabricantes que introducen la capacidad de ejecutar ciertos APIs que no estén marcados en algún estándar de la capa de perfil.

2.2.1 Máquinas Virtuales J2ME

Una Máquina Virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente. En el caso de dispositivos celulares existen bastantes maquinas virtuales entre las que destacan la Kilobyte Virtual Machine (KVM) y la Compact Virtual Machine (CVM):

- **KVM**

Corresponde con la Máquina Virtual más pequeña desarrollada por Sun. Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, aproximadamente unas 24000 líneas de código, y fue diseñada para ser:

- Pequeña, con una carga de memoria entre los 40Kb y los 80 Kb, dependiendo de la plataforma y las opciones de compilación.
- Alta portabilidad.
- Modulable.
- Lo más completa y rápida posible y sin sacrificar características para las que fue diseñada.

Las características opcionales de la KVM están definidas en módulos separados en el código fuente de la KVM.

- **CVM**

La CVM (Compact Virtual Machine) ha sido tomada como Máquina Virtual Java de referencia para la configuración *Connected Limited Configuration* (CDC) y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM. Las características que presenta esta Máquina Virtual son:

- Sistema de memoria avanzado.

- Tiempo de espera bajo para el recolector de basura.
- Separación completa de la VM del sistema de memoria.
- Recolector de basura modularizado.
- Portabilidad.
- Rápida sincronización.
- Ejecución de las clases Java fuera de la memoria de sólo lectura (ROM).
- Soporte nativo de hilos.
- Baja ocupación en memoria de las clases.
- Proporciona soporte e interfaces para servicios en Sistemas Operativos de Tiempo Real.
- Conversión de hilos Java a hilos nativos.
- Soporte para todas las características de Java2 v1.3 y librerías de seguridad, referencias débiles, Interfaz Nativa de Java (JNI), invocación remota de métodos (RMI), Interfaz de depuración de la Máquina Virtual (JVMDI).

2.2.2 Configuraciones

Una configuración consiste en un entorno de ejecución Java completo que define el entorno de ejecución básico de J2ME. Su objetivo es adecuarse a las necesidades de una familia de dispositivos con capacidades similares. Cualquier configuración está formada por tres elementos:

- Una máquina virtual Java para ejecutar el bytecode de la aplicación
- Código nativo para realizar la interfaz entre Java y el sistema operativo del dispositivo
- Un conjunto de clases Java que constituyen el entorno de ejecución.

Aunque una configuración proporcione un entorno Java completo, su conjunto base de clases es reducido y debe ser ampliado por medio de perfiles o bien por clases propias definidas por el fabricante del dispositivo. Por ejemplo, ninguna configuración define clases destinadas a implementar la interfaz de usuario sino que son proporcionadas por los perfiles.
Las configuraciones no son pensadas como una solución completa, sino como un punto de partida, una base común para crear APIs adicionales que proporcionen la funcionalidad específica para los dispositivos sobre los que vayan a ejecutarse las aplicaciones.

Actualmente están definidas dos configuraciones para J2ME: Connected Device Configuration (CDC) y Connected Limited Device Configuration (CLDC).

- **Configuración para Dispositivos Conectados (CDC)**

La Configuración para Dispositivos Conectados (CDC), definida en la especificación JSR-36 del *Java Comunity Process* (JCP), está orientada a dispositivos con cierta capacidad computacional y con memoria, además deben tener la capacidad de conexión a red, generalmente de tipo inalámbrico. CDC usa una máquina virtual Java similar a la de J2SE pero con limitaciones gráficas y de memoria. Esta máquina virtual es llamada CVM (Compact Virtual Machine).

Dispositivos de este tipo son los Palm, Pocket-PC, los Set-Top (televisión digital interactiva), mensáfonos, terminales digitales de video o fotografía, impresoras, reproductores de sonido MP3, etc.

El API CDC es un subconjunto muy amplio de J2SE. Incluyendo además todas las clases del API CLCD y algunas más. Dispone de soporte completo de la maquina virtual Java 2. Entre estas últimas clases, incluye las que forman el paquete **javax.microedition,** al que añade algunas características adicionales:

- Soporte para coma flotante.
- Soporte para procesos nativos.
- Soporte para multihilo (multithread), incluso con soporte para grupos de hilos.
- Soporte para la manipulación de sistemas de ficheros.
- Soporte para la serialización de objetos.
- Soporte para conexiones HTTP.
- Soporte para la mayoría de las colecciones del API Collections de J2SE.
- Cargador de clases definido por el usuario.
- Soporte de red, mediante el paquete **java.net**.
- Soporte para los paquetes J2SE.

El estudio de esta configuración (CDC) queda fuera del alcance de esta Tesis la cual esta orientada al estudio de la configuración Connected Limited Device Configuration (CLDC) y el perfil Mobile Information Device Profile (MIDP). Los cuales analizaremos a continuación.

- **Configuración para Dispositivos con Conexión Limitada (CLDC)**

Esta Configuración para Dispositivos con Conexión Limitada (CLDC) fue creada para poder mantener las características de Java sobre dispositivos de capacidad muy limitada. Para este dispositivo se dispone una KVM, donde Kilo corresponde a Kilobyte.

CLDC es la expresión más minimalista de J2ME, definida en la especificación JSR-30 del programa *Java Comunity Process* (JCP).

Los dispositivos que usan CLDC deben cumplir los siguientes requisitos:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución.
- Procesador de 16 o 32 bits con al menos 25 MHz de velocidad.
- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.
- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

La CLDC aporta las siguientes funcionalidades a los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su Máquina Virtual (KVM).
- Un subconjunto de las bibliotecas Java del núcleo.
- Soporte para E/S básica.
- Soporte para acceso a redes.
- Seguridad.

La tabla 2.1 nos muestra los paquetes básicos incluidos en la CLDC.

Nombre de paquete CLDC	Descripción
java.io	Clases y paquetes estándar de E/S. Subconjunto de J2SE.
java.lang	Clases e interfaces de la Máquina Virtual. Subconj. de J2SE.
java.util	Clases, interfaces y utilidades estándar. Subconj. de J2SE.
javax.microedition.io	Clases e interfaces de conexión genérica CLDC

Tabla 2.1 Librerías CLDC

Estos paquetes a excepción de javax.microedition son subconjuntos de los paquetes de J2SE **java.lang**, **java.io**, **java.util**. Es decir, que no todos los métodos soportados por el J2ME están contenidos en el API CLDC.

2.2.3 Perfiles

El perfil es un grupo más específico de APIs, desde el punto de vista del dispositivo. Es decir, la configuración se ajusta a una familia de dispositivos, y el perfil se orienta hacia un grupo determinado de dispositivos dentro de dicha familia. El perfil, añade funciones adicionales a las proporcionadas por la configuración. Por eso, a la hora de construir una aplicación se necesitan tanto las APIs del perfil del dispositivo como de la configuración. Tenemos que tener en cuenta que un perfil siempre se construye sobre una configuración determinada. De este modo, podemos pensar en un perfil como un conjunto de APIs que dotan a una configuración de funcionalidad específica.

Podemos definir la capa de perfil como un conjunto de APIs orientado a un ámbito de aplicación determinado.

El principal perfil y el más utilizado por haber sido el primero del cual se ha proporcionado una implementación es el perfil MIDP, que es un perfil basado en la configuración CLDC. Este perfil esta promulgado en la especificación JSR-37 y es mejorado en la especificación JSR-118 del *Java Comunity Process* (JCP) que incorpora al perfil conectividad mediante sockets y datagramas, soporte para protocolo HTTPS y SSL, entre otros.

Anteriormente vimos que para una configuración determinada se usaba una Máquina Virtual Java específica. Teníamos que con la configuración CDC usábamos la CVM y que con la configuración CLDC usábamos la KVM. Con los perfiles ocurre lo mismo. Existen unos perfiles que construiremos sobre la configuración CDC y otros que construiremos sobre la CLDC. Para la configuración CDC tenemos los siguientes perfiles:

- Foundation Profile.
- Personal Profile.
- RMI Profile.

Y para la configuración CLDC tenemos los siguientes:

- PDA Profile.
- Mobile Information Device Profile (MIDP).

En la Figura 2.4 se puede ver como quedaría el esquema del entorno de ejecución al completo.

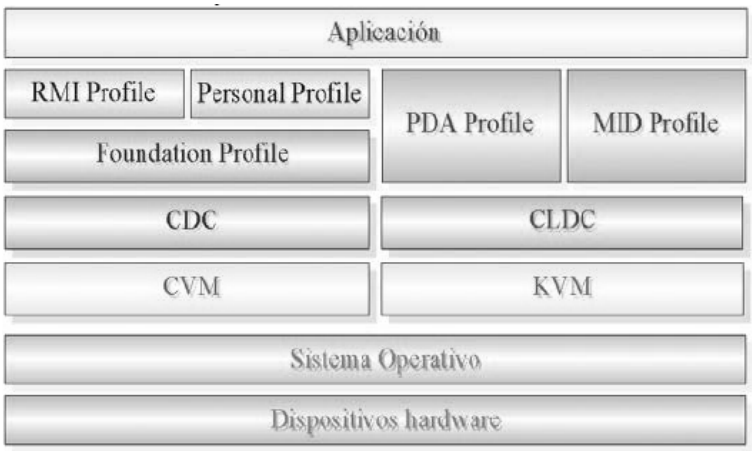


Figura 2.4 Arquitectura del entorno de ejecución de J2ME.

Un perfil puede ser construido sobre cualquier otro. Sin embargo, una plataforma J2ME sólo puede contener una configuración.

De los perfiles mencionados nos enfocaremos al Mobile Information Device Profile (MIDP) el cual utilizamos, para más información acerca de perfiles consultar [2].

Este perfil está construido sobre la configuración CLDC. Al igual que CLDC fue la primera configuración definida para J2ME, MIDP fue el primer perfil definido para esta plataforma. Este perfil está orientado para dispositivos con las siguientes características:

- Reducida capacidad computacional y de memoria.
- Conectividad limitada (en torno a 9600 bps).
- Capacidad gráfica muy reducida (mínimo un display de 96x54 pixels monocromo).
- Entrada de datos alfanumérica reducida.
- 128 Kb de memoria no volátil para componentes MIDP.
- 8 Kb de memoria no volátil para datos persistentes de aplicaciones.
- 32 Kb de memoria volátil en tiempo de ejecución para la pila Java.

- Los tipos de dispositivos que se adaptan a estas características son: teléfonos móviles, buscapersonas (pagers) o PDAs de gama baja con conectividad.

El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con:

- La aplicación (semántica y control de la aplicación MIDP).
- Interfaz de usuario.
- Almacenamiento persistente.
- Trabajo en red.
- Temporizadores.

En la tabla 2.2 podemos ver cuáles son los paquetes que están incluidos en el perfil MIDP 1.0.

Paquetes del MIDP	Descripción
javax.microedition.lcdui	Clases e interfaces para GUIs
javax.microedition.rms	*Record Management Storage*. Soporte para el almacenamiento persistente del dispositivo
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases e interfaces de conexión genérica
java.io	Clases e interfaces de E/S básica
java.lang	Clases e interfaces de la Máquina Virtual
java.util	Clases e interfaces de utilidades estándar

Tabla 2.2 Paquetes MIDP

Las aplicaciones que realizamos utilizando MIDP reciben el nombre de MIDlets (por simpatía con APPlets). Decimos así que un MIDlet es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC. En los temas siguientes nos centraremos en la creación de estos MIDlets ya que es un punto de referencia para cualquier programador de J2ME. Además, desde un punto de vista práctico MIDP es el único perfil actualmente disponible.

2.2.4 Paquetes Opcionales

Es un concepto que está muy relacionado con los perfiles, porque también proporcionan clases extra, aunque en este caso, el paquete opcional es un conjunto de APIs que proporcionan características adicionales que no pertenecen a ninguna configuración o perfil. El soporte para tecnología Bluetooth, por ejemplo, está definido en un paquete de este tipo. Incluir estos paquetes opcionales en un perfil no puede hacerse ya que ninguna de las características definidas en un perfil puede ser opcional; si un dispositivo soporta un perfil, debe soportar el perfil completo; y en el caso de Bluetooth si se incluyese en un perfil, se limitaría el uso de ese perfil solamente a los dispositivos con soporte Bluetooth.

Los paquetes opcionales también tienen su conjunto de requisitos mínimos, al igual que las configuraciones y perfiles, además de tener dependencias específicas de una configuración particular y uno o más perfiles. Los paquetes opcionales son específicos a través del *Java Comunity Process* (JCP) y disponen de su propia implementación de referencia.

2.3 MIDLETS

Ya hemos definido un midlet como una aplicación Java que cumple con la especificación MIDP. Están diseñados para ser ejecutados en dispositivos con poca capacidad gráfica, de cómputo y de memoria. En estos dispositivos no disponemos de líneas de comandos donde poder ejecutar las aplicaciones que queramos, si no que reside en él un software que es el encargado de ejecutar los MIDlets y gestionar los recursos que éstos ocupan. En el siguiente apartado vamos a hablar mas profundamente de este software que no es otro que el gestor de aplicaciones.

2.1.1 **El gestor de Aplicaciones**

El gestor de aplicaciones o AMS (Application Management System) es el software encargado de gestionar los MIDlets. Este software reside en el dispositivo y es el que nos permite ejecutar, pausar o destruir nuestras aplicaciones J2ME. El AMS realiza dos grandes funciones:

- Por un lado gestiona el ciclo de vida de los MIDlets.
- Por otro, es el encargado de controlar los estados por los que pasa el MIDlet mientras está en la memoria del dispositivo, es decir, en ejecución.

2.1.2 **Ciclo de vida de un MIDlet**

El ciclo de vida de un MIDlet pasa por 5 fases (figura 2.5) descubrimiento o localización, instalación, ejecución, actualización y borrado.

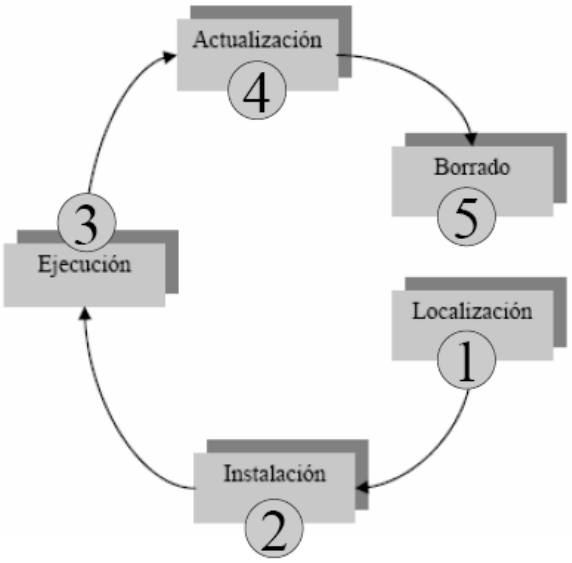


Figura 2.5 Ciclo de vida de un MIDlet.

El AMS es el encargado de gestionar cada una de estas fases de la siguiente manera:

1. Descubrimiento: Esta fase es la etapa previa a la instalación del MIDlet y es dónde seleccionamos a través del gestor de aplicaciones la aplicación a descargar. Por tanto, el gestor de aplicaciones nos tiene que proporcionar los mecanismos necesarios para realizar la elección del MIDlet a descargar. El AMS puede ser capaz de realizar la descarga de aplicaciones de diferentes maneras, dependiendo de las capacidades del dispositivo, por

ejemplo, esta descarga la podemos realizar mediante un cable conectado a un ordenador o mediante una conexión inalámbrica.

2. Instalación: Una vez descargado el MIDlet en el dispositivo, comienza el proceso de instalación. En esta fase el gestor de aplicaciones controla todo el proceso informando al usuario tanto de la evolución de la instalación como de si existiese algún problema durante ésta. Cuándo un MIDlet está instalado en el dispositivo, todas sus clases, archivos y almacenamiento persistente están preparados y listos para su uso.
3. Ejecución: Mediante el gestor de aplicaciones vamos a ser capaces de iniciar la ejecución de los MIDlets. En esta fase, el AMS tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan durante esta ejecución. Esto lo veremos un poco más en profundidad más adelante.
4. Actualización: El AMS tiene que ser capaz de detectar después de una descarga si el MIDlet descargado es una actualización de un MIDlet ya presente en el dispositivo. Si es así, nos tiene que informar de ello, además de darnos la oportunidad de decidir si queremos realizar la actualización pertinente o no.
5. Borrado: En esta fase el AMS es el encargado de borrar el MIDlet seleccionado del dispositivo. El AMS nos pedirá confirmación antes de proceder a su borrado y nos informará de cualquier circunstancia que se produzca.

Hay que indicar que el MIDlet puede permanecer en el dispositivo todo el tiempo que queramos. Después de la fase de instalación, el MIDlet queda almacenado en una zona de memoria persistente del dispositivo MID. El usuario de éste dispositivo es el encargado de decidir en qué momento quiere eliminar la aplicación y así se lo hará saber al AMS mediante alguna opción que éste nos suministre.

2.1.3 **Estados de un MIDlet en fase de ejecución**

Además de gestionar el ciclo de vida de los MIDlets, como ya hemos visto, el AMS es el encargado de controlar los estados del MIDlet durante su ejecución. Durante ésta el MIDlet es cargado en la memoria del dispositivo y es aquí donde puede transitar entre 3 estados diferentes: activo, en pausa y destruido.

Cuándo un MIDlet comienza su ejecución, está en el estado “Activo” pero, ¿qué ocurre si durante su ejecución recibimos una llamada o un mensaje? El gestor de aplicaciones debe ser capaz de cambiar el estado de la aplicación en función de los eventos externos al ámbito de ejecución de la aplicación que se vayan produciendo. En este caso, el gestor de aplicaciones interrumpiría la ejecución del MIDlet sin que se viese afectada la ejecución de éste y lo pasaría al estado de “Pausa” para atender la llamada o leer el mensaje. Una vez que terminemos de trabajar con el MIDlet y salgamos de él, éste pasaría al estado de “Destruido” dónde sería eliminado de la memoria del dispositivo. Cuando decimos que el MIDlet pasa al estado “Destruido” y es eliminado de memoria, nos referimos a la memoria volátil del dispositivo que es usada para la ejecución de aplicaciones. Una vez finalizada la ejecución del MIDlet podemos volver a invocarlo las veces que queramos ya que éste permanece en la zona de memoria persistente hasta el momento que deseemos desinstalarlo.

Un MIDlet durante su ejecución pasa por 3 estados diferentes. Como ya hemos visto en el apartado anterior, estos tres estados son:

- Activo: El MIDlet está actualmente en ejecución.
- Pausa: El MIDlet no está actualmente en ejecución. En este estado el MIDlet no debe usar ningún recurso compartido. Para volver a pasar a ejecución tiene que cambiar su estado a Activo.
- Destruido: El MIDlet no está en ejecución ni puede transitar a otro estado. Además se liberan todos los recursos ocupados por el MIDlet.

La Figura 2.6 nos muestra el diagrama de estados de un MIDlet en ejecución:

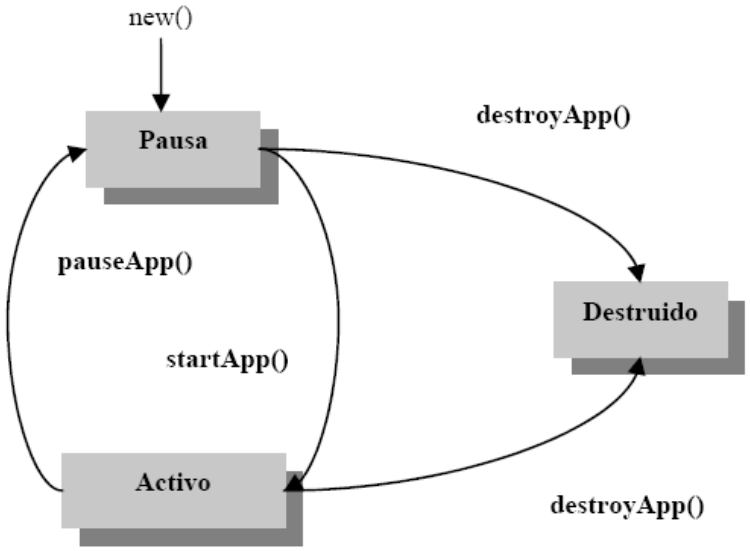


Figura 2.6 Estados de un MIDlet

Como vemos en el diagrama, un MIDlet puede cambiar de estado mediante una llamada a los métodos MIDlet.startApp(), MIDlet.pauseApp() o MIDlet.destroyApp(), los cuales están contenidos en el paquete **javax.microedition.midlet**. El gestor de aplicaciones cambia el estado de los MIDlets haciendo una llamada a cualquiera de los métodos anteriores. Un MIDlet también puede cambiar de estado por sí mismo.

2.1.4 **Estructura de los MIDlets**

En este punto ya sabemos cuáles son los estados de un MIDlet, conocemos su ciclo de vida. Ahora vamos a ver cuál es la estructura que comparten todos los MIDlets.

Hemos de decir que los MIDlets, al igual que los applets carecen de la función main(). Aunque existiese, el gestor de aplicaciones la ignoraría por completo. Los MIDlets tienen la siguiente estructura:

```
import javax.microedition.midlet.*
public class MiMidlet extends MIDlet
   public MiMidlet() {
   /* Éste es el constructor de clase. Aquí debemos
        inicializar nuestras variables.
   */
   }
   public startApp(){
```

```
    /* Aquí incluiremos el código que queremos que el
        MIDlet ejecute cuándo se active.
    */
    }
    public pauseApp(){
    /* Aquí incluiremos el código que queremos que el
    MIDlet ejecute cuándo entre en el estado de pausa
    (Opcional)
                            */
    }
    public destroyApp(){
    /* Aquí incluiremos el código que queremos que el
        MIDlet ejecute cuándo sea destruido. Normalmente
    aquí se liberaran los recursos ocupados por el
    MIDlet como memoria, etc. (Opcional)
                        */
    }
}
```

3 MATLAB

MATLAB es el nombre abreviado de “MATrix LABoratory”, es un programa de cálculo científico y técnico que permite realizar cálculos numéricos con vectores y matrices, además permite realizar varios tipos de gráficos en dos o tres dimensiones.

Para ciertas aplicaciones es muy rápido, cuando puede ejecutar sus funciones en código nativo con los tamaños más adecuados para aprovechar sus capacidades de vectorización. En otras aplicaciones resulta bastante más lento que el código equivalente desarrollado en C/C++ o Fortran pero es una herramienta de alto nivel para aplicaciones técnicas y científicas.

MATLAB cuenta con un código base así como de librerías llamadas toolboxes.

3.1 Entorno de trabajo MATLAB

Entre las componentes más importantes que se encuentran en su entorno de trabajo son las siguientes:

1. Matlab Desktop o el Escritorio de Matlab, es la ventana que contiene todos los componentes, figura 3.1.

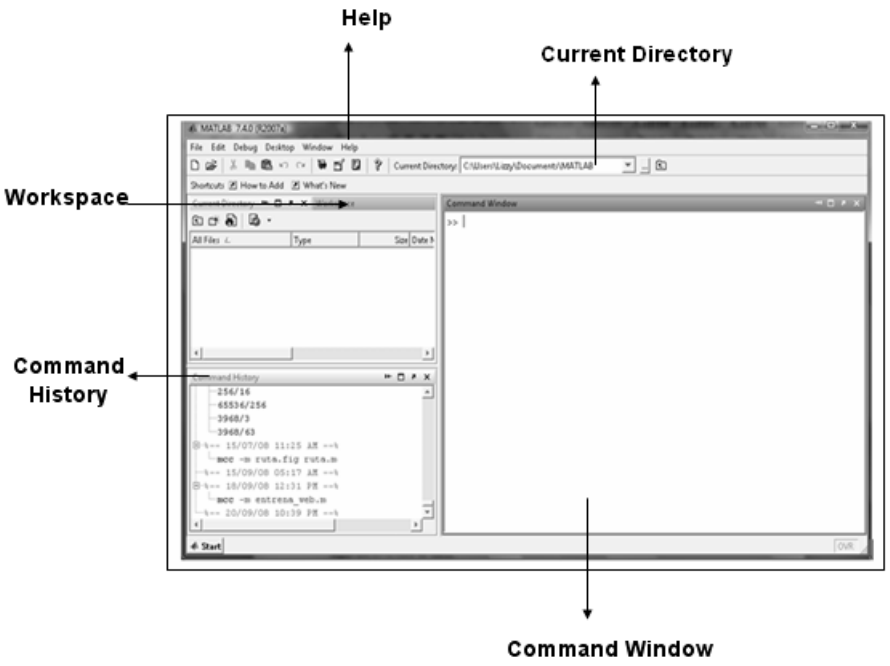


Figura 3.1 Escritorio de MATLAB

2. Entre los componentes individuales del escritorio de Matlab, podemos mencionar:

    a. Command Window o ventana de comandos
    b. Command History o ventana histórica

c. Workspace Browser o espacio de trabajo
d. Current Directory Browser o directorio actual
e. Help Browser o ventana de ayuda

3.2 El Editor

MATLAB tiene unos ficheros-M, los cuales son de texto ASCII con la extensión *.m, los cuales contienen conjuntos de comandos o funciones, para ejecutarlos es necesario teclear su nombre en el Command Window.

MATLAB cuenta con un editor el cual permite crear y modificar los ficheros, así como ejecutarlos. El Editor muestra con diferentes colores los diferentes elementos consecutivos de los comandos.

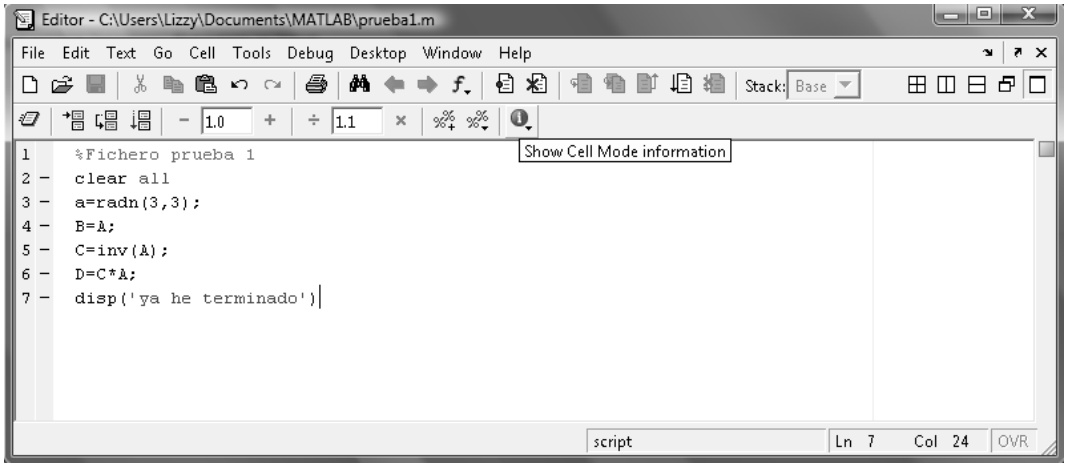


Figura 3.2 Editor de MATLAB

3.3 Manejo de variables

Una variable es un nombre que se le asigna a una entidad numérica que en el caso de MATLAB puede ser una matriz, un vector o un escalar.

Una expresión de MATLAB puede tener dos formas, una de ellas es asignando el nombre a una variable:

```
>> variable=expresion
o evaluando el resultado
>> expresion
```

en este caso el resultado se asigna automáticamente a una variable interna de MATLAB llamada ans, la cual almacena el último resultado obtenido.

Del mismo modo que C, MATLAB distingue entre mayúsculas y minúsculas en los nombres asignados a las variables.

Los nombres de las variables tampoco pueden comenzar por un número aunque si puede contenerlos, tampoco pueden contener operadores ni puntos.

En la pestaña Worspace podernos observar los nombres de las variables así como su valor.

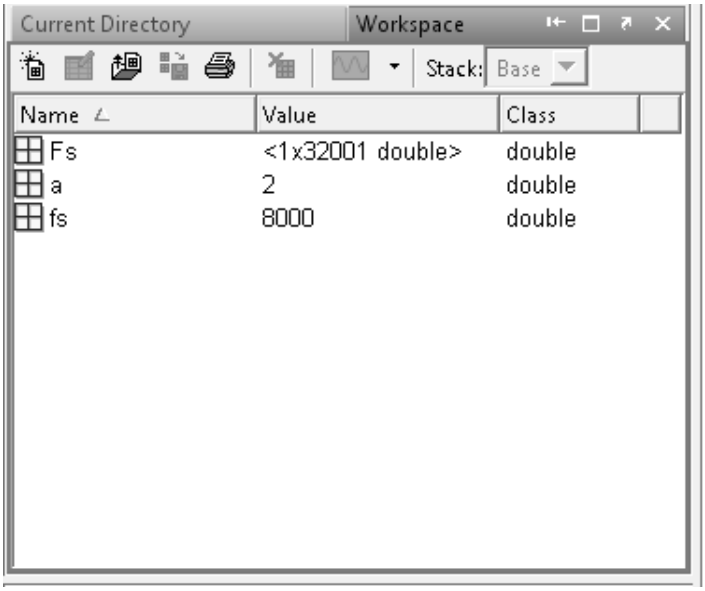


Figura 3.3 Pestaña Workspace

3.4 Manejo de funciones

Una función es un programa especial que pide datos de entrada y nos devuelve otros datos como salida. MATLAB posee un gran número de funciones incorporadas, definidas en ficheros *.m, estas extienden las posibilidades de programar en MATLAB.

Del mismo modo que en C, una función en MATLAB tiene nombre, valor de retorno y argumentos. Una función se le llama solo utilizando su nombre en una expresión o usándolo como un comando. Por ejemplo:

```
>> a=cos(90)-sin(45);
```

Otra de las características de MATLAB es que las funciones no tiene argumentos no llevan paréntesis. Los nombres de las funciones de MATLAB no son palabras reservadas del lenguaje, podemos llamar a una variable cos pero para ello debemos eliminar las variables del mismo nombre.

Es importante recordar, que en un programa las variables son de uso local, es decir, su nombres no afectan al resto de las variables en MATLAB.

Existen diversos tipos de funciones, los más usuales son:

1. Funciones matemáticas elementales.
2. Funciones especiales.

3. Funciones matriciales elementales.
4. Funciones matriciales específicas.
5. Funciones para la descomposición y/o factorización de matrices.
6. Funciones para análisis estadístico de datos
7. Funciones para análisis de polinomios.
8. Funciones para integración de ecuaciones diferenciales ordinarias.
9. Resolución de ecuaciones no-lineales y optimización.
10. Integración numérica.
11. Funciones para procesamiento de señal.

Existe una dualidad entre comandos y funciones, un comando se puede escribir como:

>> comando arg 1 arg2

la cual es equivalente a la siguiente función

>> comando ('arg 1','arg2')

Esta característica es útil al programar porque permite crear argumentos con operaciones propias.

Para crear ficheros podemos hacer uso del editor de MATLAB, llamamos al archivo *.m como a nuestra función y en el primer renglón del archivo debe contener la sintaxis de una nueva función:

>> function [out1, out2, ...] = funname(in1,in2, ...)

Colocando las entradas se puede compilar el programa en el mismo editor o bien desde el Command Window darle la o las entradas y desde ahí compilar.

Las funciones para manipular ficheros son las siguientes:

- **fopen:** sirve para abrir un archivo que luego se utilizará para lectura, escritura o ambos. La forma general de esta función es:

  [fi, mensaje] = fopen(’archivo’, ’opciones’)

  Las opciones a seleccionar pueden ser las siguientes:

  ’r’: lectura (read)
  ’w’: escritura sobreescribiendo si es que el archivo existía (write)
  ’a’: escritura pero añadiendo a lo ya existente si es que el archivo existía (add)
  ’r+’: lectura y escritura

- **fclose:** cierra los archivos indicados en el paréntesis, puede ser uno o todos.

  fclose('all')
  fclose('reconoce.m')

- **fprintf:** escribe archivos de texto formateados.

fprintf(fid, format, A, ...)

donde fid es el archivo abierto por fopen, y posteriormente se escribir el formato en el cual deseamos verlo por ejemplo %d (notación decimal), %s (cadena de caracteres), etc.

3.5 Manejo de archivos

3.5.1 Archivos *.m

Todos los comandos o funciones creadas se pueden utilizar directamente desde la línea de el Command Window pero con un archivo *.m el programa puede ser ejecutado más rápido además podemos modificarlo una y otra vez sin tener que teclear todo nuevamente.

3.5.2 Archivos *.wav

Para poder capturar una señal de audio por la tarjeta auxiliar de sonido MATLAB posee la función wavrecord(n,Fs,ch) donde n es el tiempo en segundos, fs la frecuencia de muestreo(los estándares son 8000,11025,22050y 44100) y ch el tipo de canal (1 para mono y 2 para estéreo).

Para guardar la señal que se capturo se usa la función wavwrite(y,Fs,N, 'filename.wav'), donde y son los datos a escribir, Fs es la frecuencia de muestreo(por default MATLAB toma 11025),N es el número de bits (pueden ser 8,16,24 o 32) y filename es el nombre del archivo donde se guardaran los datos de y pero en formato wav.

Para poder escuchar el archivo *.wav, este debe ser asignado a una variable usando el comando wavread('filename.wav'), para después reproducirlo con la función sound ('filename.wav').

3.5.3 Manejo de archivos *.mat

Los archivos *.mat, son usados para salvar datos del disco C pero también contiene un mecanismo para mover datos de MATLAB entre plataformas y para importar y exportar datos para las aplicaciones independientes de MATLAB. Esto simplifica el uso de los archivos *.mat en aplicaciones fuera de MATLAB, podemos desarrollar una librería de acceso a rutinas con el prefijo mat que podemos usar en programas de C o de Fortran para leer y escribir archivos *.mat.

- Se puede guardar el espacio de trabajo con el comando save que nos genera u archivo *.mat y con el comando load obtenemos de nuevo esas variables, es una manera de importar y exportar.
- La instrucción diary graba tanto los comandos introducidos como la salida de MATLAB, pero no graba los valores de las variables y matrices.

3.6 Función mcc

El compilador de MATLAB se llama mbuild este se invoca únicamente cuando es necesario configurarlo o bien cuando deseamos compilar archivos C para obtener un ejecutable. Para el resto de las aplicaciones se usa la función mcc que también invoca al compilador de MATLAB.

Las opciones más importantes son:

- **Creación de un archivo MEX(subrutinas creadas en C o en Fortran):** para cambiar un archivo matlab (.m) a C y crear el archivo CMEX se usa el comando:

  mcc -x filename.m

- **Creación de una función S de Simulink:** para traducir una archivo .m a C y crear la función S de Simulink se invoca al compilador con la siguiente instrucción:

  mcc -S filename.m

- **Creación de una aplicación independiente C:** para cambiar un archivo .m a C y generar directamente el ejecutable independiente que puede ser llamado sin tener instalado MATLAB se usa la siguiente instrucción:

  mcc -m filename.m

- **Creación una aplicación independiente C++:** Se usa la siguiente instrucción:

  mcc -p filename.m

- **Creación de una aplicación independiente C con funciones de librerías gráficas:** para convertir archivos .m que contienen funciones de tratamiento de gráficos a C se utiliza la siguiente instrucción:

  mcc -B sgl filename.m

3.7 Creación de interfaces

La GUIDE (Graphical User Interfase Development Environment), figura 3.4, es un set de herramientas de MATLAB que permiten crear GUIs (Graphical User Interfaces) de una manera muy sencilla al usuario y reducen la labor al grado de seleccionar, tirar, arrastrar y personalizar propiedades.

Ya que se encuentran todos los elementos de control se pueden editar con las funciones Callback de cada uno de estos con tan solo escribir código de MATLAB que se ejecutará cuando se utilice la aplicación.

La cadena de invocación es una cadena que consiste en un comando, un conjunto de órdenes. En esta cadena se especifican todas las tareas que deben ejecutarse. Las instrucciones pueden ser un solo comando o varios. Todos los detalles de los cálculos se pueden escribir en un archivo *.m.

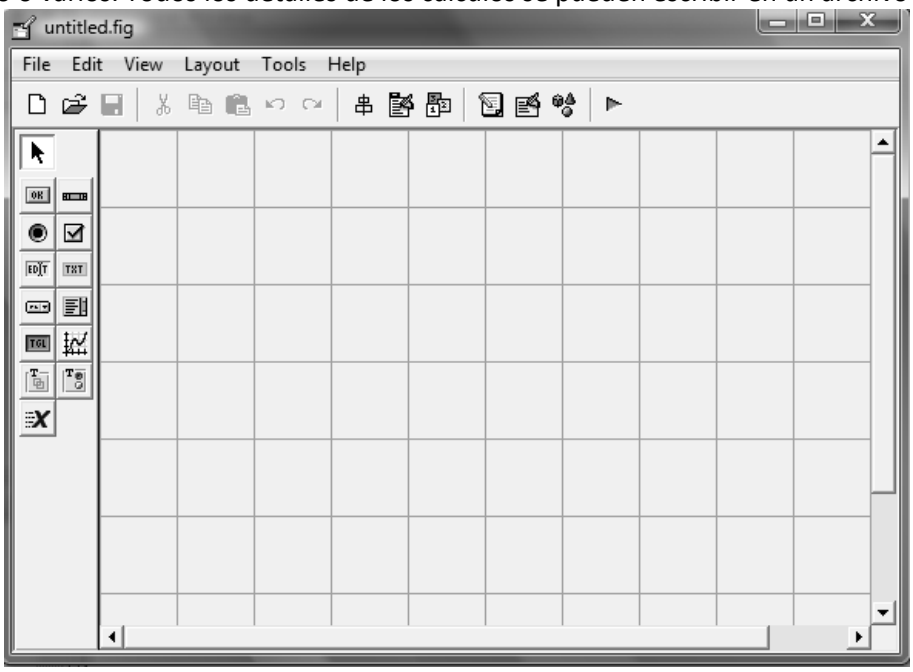


Figura 3.4 Ventana de GUIDE

Los elementos principales del GUIDE son:

- **Barra de Menús:** en ella se localizan las funciones elementales de edición de GUI's y se localiza en la parte superior de la ventana del GUIDE.
- **Paleta de Componentes: en** ella se encuentran los uicontrols que nos permiten seleccionar los objetos y se localiza en la parte lateral izquierda.
- **La Barra de Herramienta:** En ella se encuentran los siguientes botones:

a) **Botón de ejecución:** cuando se presiona se crea la interfaz, es similar a correr el programa.

b) **Alineación de Componentes:** esta opción permite alinear los objetos que se encuentra en el área de trabajo.

c) **Propiedades del Inspector:** al presionar este icono podemos asignar y modificar las propiedades de los objetos de forma personalizada.

d) **Navegador de Objetos:** al dar click sobre este botón vemos los objetos que definimos pero en forma de árbol y además se pueden seleccionar los objetos.

e) **Editor de Menús:** El redactor de Menú crea menús de ventana y menús de contexto. La Interfaz de Grafica de Usuario (GUI) se crea en una ventana de figura que consta de los siguientes componentes:

1. Menú de interfaz con el usuario.
2. Dispositivo de control de interfaz con el usuario.
3. Ejes para exhibir graficas o imágenes

4 PROGRAMACIÓN EN EL SERVIDOR APACHE Y PHP

4.1 Introducción a Apache Server

El término servidor describe una PC que contiene software que realiza tareas específicas por los usuarios, es un administrador de datos que los comparte con un grupo de dispositivos por medio de una red.

Un servidor web se refiere al software que funciona en una PC y que maneja la entrega de los componentes de las páginas web como respuesta a peticiones de los navegadores de los clientes mediante el uso del protocolo HyperText Transfer Protocol *(*HTTP). Los archivos necesarios para montar un sitio web son almacenados en el servidor web, el cual permite a los usuarios acceder a los archivos y servicios que la web soporta.

- **Servidores de Aplicaciones (Application Servers):** Opera como un tipo de middleware (software que conecta dos aplicaciones), este tipo de servidores proveen un enlace entre dos o más servicios diferentes que arrojan como resultado una página web en un navegador. El uso mas común es la interconexión de lenguajes de programación para sitios web dinámicos como PHP con bases de datos como MySQL.

- **Apache Server:** es un servidor HTTP libre de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y NetWare, e implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Herramientas como Apache Server son fácilmente instalables y configurables de acuerdo a las necesidades del servicio web.

  La arquitectura del servidor Apache es modular, es decir, que las tareas que realiza están agrupadas en bloques que realizan funciones específicas. A diferencia de los servidores monolíticos, donde una sola estructura realiza todos los procesos, el servidor apache consta de una sección núcleo (*core*) o módulo básico y diversos módulos que proporcionan funcionalidad y robustez necesarias y básica para un servidor web.

La arquitectura de un servidor apache se muestra en la figura 4.1:

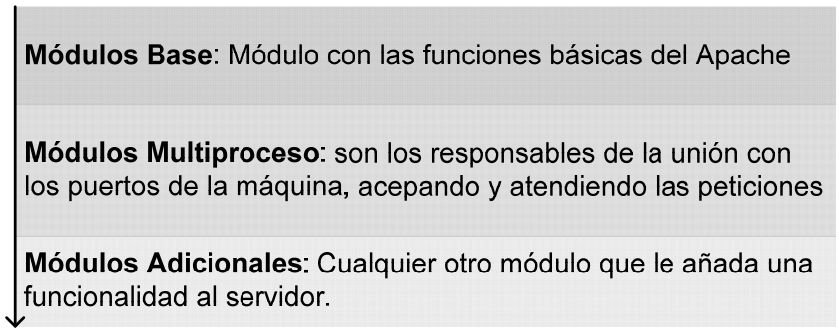


Figura 4.1 Arquitectura del servidor Apache

Módulo Base.

Este módulo implementa las funciones básicas del servidor, adicionalmente realiza otras funciones que permiten la interacción más simple entre el módulo base y los otros módulos.

Los componentes del módulo núcleo en conjunto realizan funciones como la comunicación directa entre cliente-servidor mediante el protocolo HTTP, el flujo de procesamiento requerido entre el núcleo y los diferentes módulos en el orden indicado, tener un orden de tareas asignadas y seguirlas, lectura de archivos de configuración y soporte del "virtual host".

Un módulo implementa solo ciertas funciones durante el servicio hacia un cliente, por lo tanto es necesario más de un módulo para completar la respuesta a una petición. Ninguno de los módulos interactúa con otro directamente, la transferencia de información se realiza a través del módulo núcleo.

Módulos Multiproceso.

La finalidad de estos módulos es incrementar la funcionalidad del servidor apache. Los módulos tienen la misma interfaz con el núcleo del servidor.

El servidor apache permite cargar módulos dinámicamente cuando estos son requeridos, una vez que estos son cargados las funciones y métodos que contiene cada modulo pueden ser empleados.

Apache maneja el concepto de "handler", el cual es para apache una guía de las acciones que debe de realizar para responder una petición en sus múltiples niveles. Los "handlers" son establecidos por cada módulo y pueden estar presentes o no en las fases de una petición. Cada uno de los "handlers" es llamado de acuerdo a la fase que se esté realizando en el proceso de responder una petición.

La razón de utilizar "handlers" es que nos permiten guardar información entre las diferentes peticiones y usar la información de una petición en otra. Existen funciones provenientes de los módulos que están relacionadas con la configuración del servidor y que se cargan cuando el servidor es inicializado.

Módulos Estándar.

El servidor apache cuenta con módulos estándar que lo proveen de una completa funcionalidad como servidor web. Las funciones que realizan estos módulos son: establecer las rutas correctas de acuerdo al usuario, identificar las direcciones URL, autentificar y autorizar las diferentes fases de las peticiones, determinar el tipo de objeto solicitado al servidor, regresar una respuesta correcta al cliente y autentificación de la solicitud (loggin).

Virtual Host.

El servidor apache puede responder a más de un nombre, (www.tesis o www2.tesis), cada uno asignado a una diferente dirección IP por la computadora. El conjunto de direcciones IP pueden ser asignadas a una red física o a una virtual. Apache es capaz de decidir bajo que acceso IP esta trabajando y tener diferentes configuraciones para cada uno.

4.2 Introducción a PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación de "código abierto", interpretado, de alto nivel, embebido en páginas WEB dinámicas y ejecutado en un servidor. Es usado principalmente en interpretación, del lado del servidor (server-side scripting)[1], pero puede ser utilizado desde una interfaz de línea de comandos o en aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK. [2]

En vez de escribir un programa con muchos comandos para crear una salida en HTML, se escribe el código HTML con cierto código PHP embebido (incluido) en el mismo, que producirá cierta salida. El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de otras tecnologías multiplataforma como Javascript, en la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor, el cliente solamente recibe el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX [3], Solaris y OpenBSD [4]), Microsoft Windows, Mac OS X, RISC OS. PHP soporta la mayoría de servidores web, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI (Common Gateway Interface), PHP puede usarse como procesador CGI.

De modo que, con PHP se tiene la libertad de elegir el sistema operativo y el servidor de manera indistinta. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos. En la versión actual de PHP, muchas bibliotecas y aplicaciones grandes (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.

---

[1] El usuario realiza solicitudes a un servidor que las procesa y transforma en una página web.
[2] Bibliotecas para desarrollar interfaces gráficas de usuario.
[3] Versión de Unix desarrollada y mantenida por Hewlett Packart
[4] Sistema Operativo Libre tipo Unix

PHP cuenta con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), así que puede conectarse a cualquier base de datos que soporte tal estándar.

4.2.1 Manejo de variables en PHP

En el lenguaje PHP las variables se representan con un signo de $ seguido del nombre de la variable. Este nombre puede es sensible a las mayúsculas y minúsculas. Cada variable debe de seguir reglas para su definición, como iniciar con un carácter alfanumérico o un guión bajo.

Las variables se asignan de forma predetermina por su valor, esto es que cuando una expresión es asignada a una variable, el valor de la expresión se copia totalmente a la variable.

Este tipo de asignación de valores se denomina *expresión* y se definen como la asignación de un valor a una entidad PHP. Las expresiones básicas son las constantes y las variables.

El lenguaje PHP soporta tres tipos escalares: enteros, punto flotante y cadenas (indivisibles). PHP también soporta dos tipos compuestos (no escalares): arreglos y objetos. Se puede asignar cada uno de estos tipos de valor a variables o bien retornarse de funciones.

Las expresiones también pueden tener un valor booleano, lo que implica que la variable es *true* o *false*, y que obtienen este tipo de valores de una condición o ciclo.

Existe otro método para asignar valores a las variables y es la *asignación por referencia*, lo que implica que una nueva variable se convierte en un alias o que apunta a la variable original. La sintaxis para hacer este tipo de asignación es $var2 = &$var1, donde el signo de ampersand indica que la variable 2 es referenciada a la variable 1. Únicamente las variables con nombre pueden ser referenciadas por referencia.

4.2.2 Manejo de funciones en PHP

Una función es un conjunto de instrucciones que procesan expresiones para obtener un resultado y que pueden ser ejecutadas en cualquier parte del código. La sintaxis para definir una función es la siguiente:

```
<?php
      function nombre_de_la_funcion ($arg_1, $arg_2,..., $arg_n)
      {
            Cuerpo de la función
      }
?>
```

Cualquier instrucción válida del lenguaje PHP puede aparecer en el cuerpo de la función, esto incluye otras funciones y manipulación de clases (definición). Dependiendo del propósito de la función y de la versión de PHP con la que se trabaje, se define el número de argumentos.

Los parámetros de una función se pasan por valor de manera que si cambia el valor del argumento dentro de la función, dicho valor no se ve modificado fuera de ella. Para que una función modifique el valor de sus parámetros debemos asignar dichos parámetros por referencia. El procedimiento de asignación de argumentos por referencia consiste en asignar argumentos de la siguiente manera a una función:

```
function nombre_de_la_funcion (&$arg_1, &$arg_2,..., &$arg_n)
{
      Cuerpo de la función
}
```

Una vez que se hace referencia a los argumentos antecedidos por un signo &, la función modificara dicha variable y el resultado de esta modificación se podrá apreciar fuera de la función.

Las funciones regresan valores de acuerdo a la finalidad para la cual fueron definidas, pueden devolver cualquier tipo de valor incluyendo listas y objetos. La sintaxis para tal efecto es simple, el operador “return” seguido del valor, lista u objeto a será entregado como resultado de la función.

4.2.3 Manejo de archivos en PHP

La librería de manejo de archivos en PHP es nativa y pertenece al núcleo del lenguaje, por lo que no se requiere instalación de algún componente extra. Las directivas de las funciones para manejo de archivos están declaradas en el archivo de configuración php.ini. Las directivas en su conjunto permiten manipular los archivos y realizar las siguientes funciones:

- Ejecutar comandos de apertura de archivos del tipo URL, es decir, acceso a objetos de manera remota mediante los protocolos http o ftp.
- Definición del tipo de usuario que define php.
- Definición de contraseñas para el protocolo de transferencia de archivos.
- Lectura de datos para interpretar el lenguaje de procedencia (UNIX, MS-Dos o Macintosh).

Las funciones básicas utilizadas en el desarrollo de la tesis son las siguientes:

- **fopen.** Abre un archivo local o remoto. Asocia un recurso con nombre especificado por su primer parámetro. Si el parámetro de nombre es definido como un archivo local, PHP intenta abrir la secuencia del archivo. El archivo debe tener los permisos de acceso que requiere php para actuar sobre él.

  Si el parámetro nombre es identificado como una dirección URL PHP verifica que la directiva de permiso allow_ url_fopen se encuentre habilitada para poder operar sobre el archivo.

  Existe un segundo parámetro que permite acceder a un archivo cambiando la forma en que se realiza este acceso, el parámetro *modo* especifica el tipo de acceso que requiere para la secuencia asignada en el primer parámetro y puede asignar cualquiera de las siguientes formas de apertura:

- Apertura para leer exclusivamente el archivo.
- Apertura para lectura y escritura.
- Apertura para solo escritura.
- Creación del archivo y escritura o lectura.

Una vez abierto el archivo la variable definida con fopen tendrá un apuntador de archivo o un valor *false* si el archivo no se pudo abrir.

- **fclose.** Cierra el apuntador a un archivo abierto que está definido por una variable $puntero. Esta instrucción solo es válida cuando se ha abierto un archivo de manera exitosa con fopen.

- **fgets.** Esta instrucción nos permite leer una línea desde el apuntador de un archivo abierto mediante fopen. Esta instrucción depende de dos argumentos, el primero debe ser una variable tipo apuntador a un archivo abierto con éxito. El segundo argumento es la longitud de caracteres que serán leídos por línea, la lectura termina cuando se obtiene la longitud de -1 bytes, o cuando se alcanza un salto de línea o cuando se llega al final del archivo.

4.3 Manejo de Bases de Datos en PHP

4.3.1 Introducción a MySQL

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Existen programas denominados sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

MySQL es un sistema de gestión de bases de datos, una base de datos es conjunto estructurado de datos. Para añadir, acceder, y procesar los datos almacenados en una base de datos, se necesita un sistema de gestión de base de datos como MySQL Server. El manejo de grandes cantidades de datos y los sistemas de gestión de bases de datos desempeñan una función fundamental en el manejo de información electrónica, son usados como aplicaciones autónomas o como parte de otras aplicaciones.

Una base de datos relacional como lo es MySQL, almacena datos en tablas separadas en lugar de mantener todos los datos en un solo contenedor. Esto añade velocidad y flexibilidad. La parte SQL de "MySQL" se refiere a "Structured Query Language". SQL (Structured Query Language),es el lenguaje estandarizado más común para acceder a bases de datos y está definido por el estándar ANSI (American National Standards Institute)/ISO (International Organization for Standardization) SQL.

MySQL es un sistema cliente/servidor que consiste en un servidor SQL multi-hilo que trabaja con diferentes terminales, programas y bibliotecas cliente, herramientas administrativas y una amplia gama de interfaces de programación para aplicaciones (APIs).

4.3.1.1 Características de MySQL

Algunas de las características más importantes de este sistema de gestión de bases de datos que se han explotado en el desarrollo de este trabajo son:

Diseño y Portabilidad

Este sistema fue creado en C/C++ y es multiplataforma, tiene APIs para poder interactuar con lenguajes de programación como C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl, además puede ser ejecutado como un programa aislado para entornos de red cliente/servidor, también puede ser usado como biblioteca o conectarse en aplicaciones autómatas.

Sentencias y funciones

El sistema MySQL soporta estándares SQL y ODBC, cuenta con soporte para funciones de consulta, agrupación, administración de la información, muestra de información de la misma base de datos y mezcla de distintas bases de datos en una misma línea de ejecución.

Conectividad

Los clientes se pueden conectarse mediante sockets TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) en cualquier plataforma. Los servidores y clientes interactúan con conexiones compartidas.
La interfaz para el conector ODBC proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity).

4.3.2 MySQL y PHP

La extensión de base de dato MySQL puede ser configurada y compilada por PHP, esto se hace accediendo a la biblioteca del cliente de MySQL; las versiones más modernas de PHP ya tienen habilitado el soporto para bases de datos de este tipo.

El archivo de configuración php.ini contiene mucha información de conexión y operatividad de la base de datos MySQL, las directivas de configuración de la base de datos que se definen en este archivo realizan las siguientes funciones:

Configuran las conexiones persistentes (enlaces que no se cierran cuando termina la ejecución del archivo de comandos) con MySQL, el número máximo de estas conexiones, el número máximo de conexiones por proceso, número de puerto TCP (Protocolo de Control de Transmisión) predeterminado para usar cuando existe una conexión con el servidor de bases de datos, el

nombre del socket [5] predeterminado a ser usado cuando se realicen conexiones con un servidor de bases de datos local, la conexión de usuarios a la base de datos, las contraseñas de conexión a la base de datos y el tiempo de espera máximo de conexión.

Existen dos tipos de recursos explotados durante el uso de MySQL, el primero es el identificador de enlace para una conexión con la base de datos, y el segundo es el que se explota durante las consultas, figura 4.2.

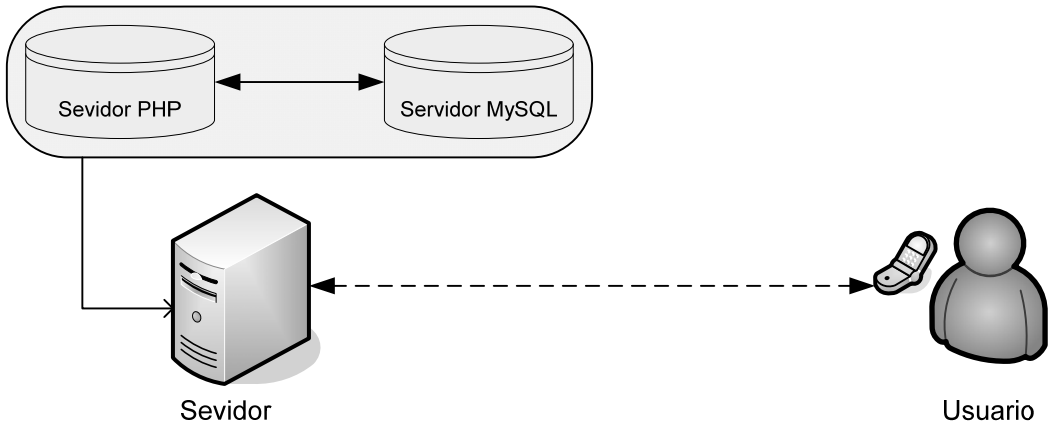


Figura 4.2 El esquema básico de conexión entre el servidor PHP y el servidor MySQL interactuando con el usuario.

[5] Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Está definido por una dirección IP, un protocolo y un número de puerto.

5 DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIÓN

El sistema de comunicación consta de una serie de peticiones al servidor desde el dispositivo móvil estas pueden ser autenticación de usuario, entrenar un nuevo comando de voz, reconocer un comando o ejecutar un comando por medio de un clic. En este capítulo explicaremos en detalle cada una de estas peticiones.

5.1 Autenticación

Debido a que el reconocedor de voz es dependiente del locutor decidimos utilizar autenticación en el servidor, una vez autenticado se abre una sesión con la configuración personal de cada usuario. En la figura 5.1 podemos observar el mecanismo de autenticación entre el dispositivo y el servidor, el usuario introduce el login, password y el destino en el servidor, al recibir la información el servidor consulta en la base de datos el login y password si estos coinciden con los almacenados entonces tendremos acceso a la aplicación de lo contrario observaremos un mensaje de error en la autenticación.

Es importante resaltar que la aplicación en el móvil envía y recibe la información por medio del protocolo HTTP utilizando el método POST (para más información consulte el apéndice C). Debido a esto la aplicación es independiente del tipo de conexión a red que utilicemos, para realizar las pruebas utilizamos los celulares modelo W610i y el Z610i de Sony Ericsson los cuales soportan el protocolo HTTP 1.1 y utilizamos una conexión a red por medio de Bluetooth.

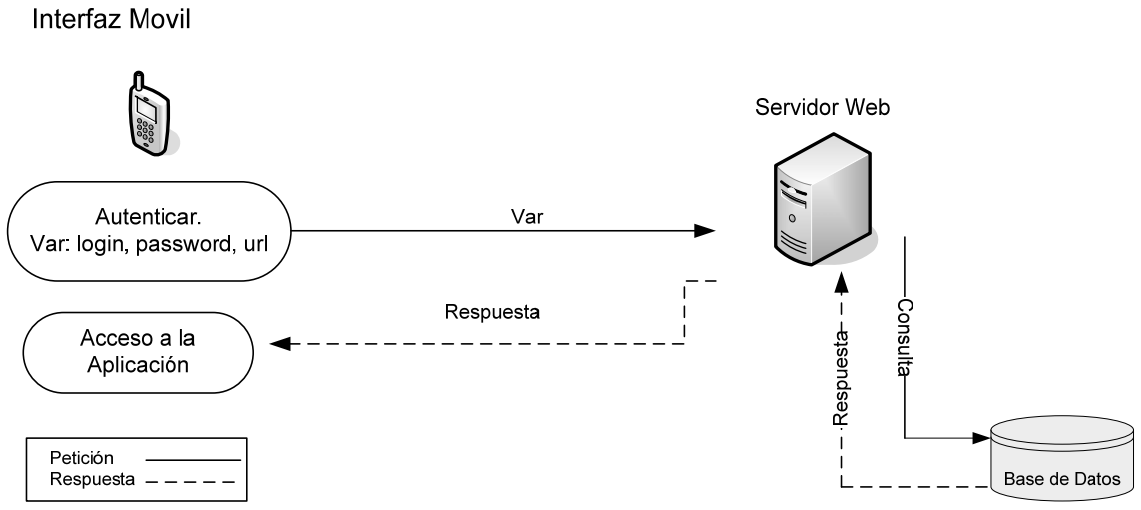


Figura 5.1 Diagrama de autenticación

5.2 Entrenamiento

Para entrenar una palabra, el servidor necesita los siguientes parámetros: el nombre que se asocia a dicha palabra, la dirección destino del servidor (URL- Uniform Resource Locator) y siete repeticiones de la palabra a entrenar. Cada vez que se ha llevado a cabo una repetición, el audio es enviado a un servidor y almacenado, dicho proceso se repite hasta recibir la séptima señal, que dispara el entrenador como se observa en la figura 5.2.

Para obtener la señal de voz utilizamos el Mobile Media API de Java Micro Edition descrito en la especificación JSR 135 [6]. Como ya mencionamos utilizamos celulares marca Sony Ericsson los cuales utilizan el formato Adaptive Multi Rate (AMR) [7] para grabar voz, este formato es un compresor de voz muy utilizado debido a que presenta una buena calidad y una alta tasa de compresión. Para acoplar la información AMR a cada una de las plataformas (Matlab y PHP), se realizaron conversiones AMR-WAV sobre Matlab y AMR -ASC en PHP.

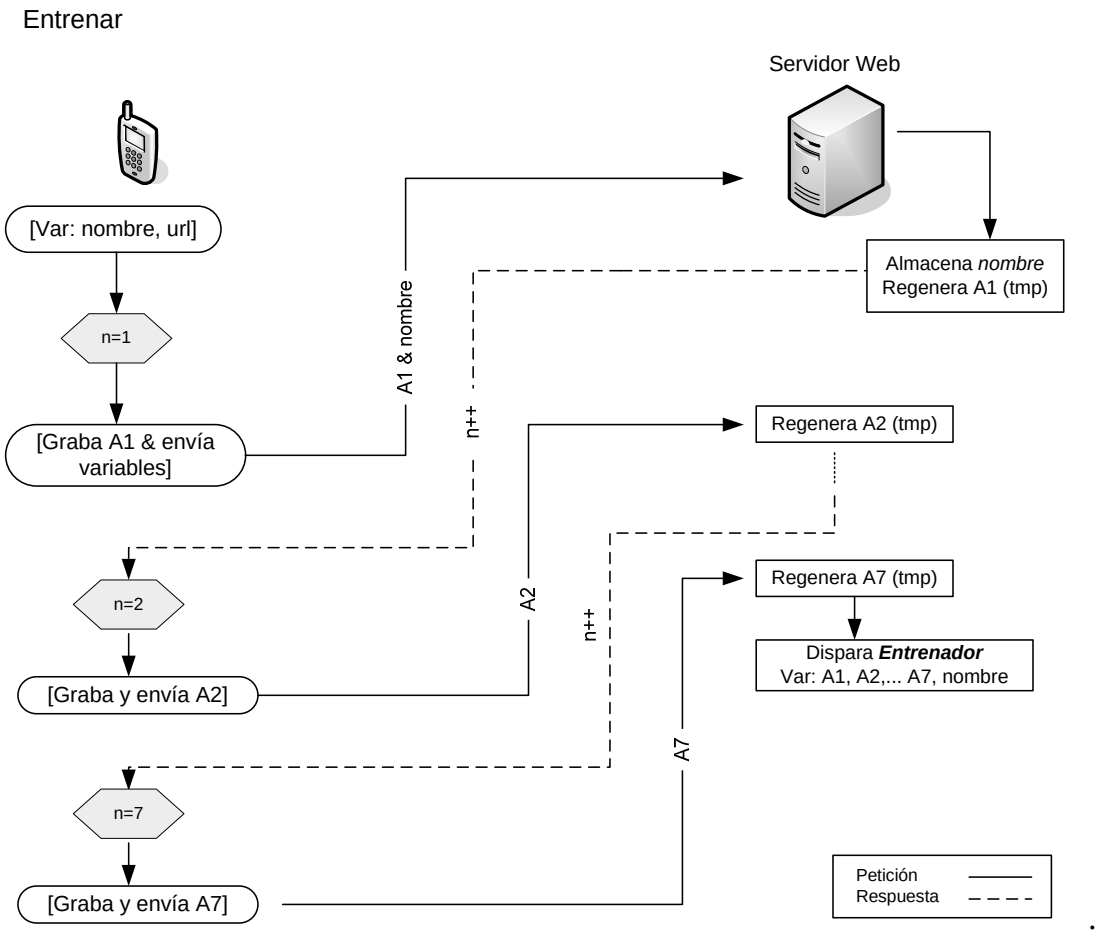


Figura 5.2 Diagrama de entrenamiento

Al finalizar el proceso de entrenamiento el servidor guarda el nombre del nuevo comando así como los centroides resultantes de la cuantización vectorial de las 7 señales.

5.2.1 Entrenador

Una vez que las señales de voz han sido enviadas al servidor, éste ejecuta el proceso de entrenamiento. Se puede describir este proceso en dos etapas, la primera se encarga de adecuar la señal en tiempo y frecuencia para ser procesada posteriormente por los algoritmos de Cuantización Vectorial (VQ). Los algoritmos de preprocesamiento se aplican a todas las señales de manera independiente. Los algoritmos de procesamiento implican el uso de todas las señales al mismo tiempo.

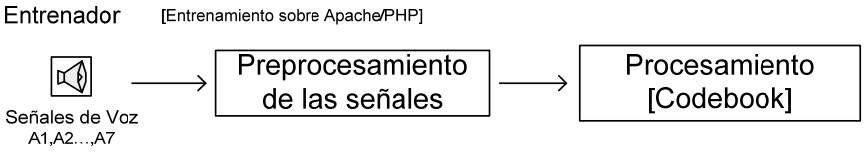


Figura 5.3 Proceso de entrenamiento

El preprocesamiento de las señales o fase de análisis, es un conjunto de procesos que se realiza sobre cada una de las señales con el propósito de adecuar sus características temporales y frecuenciales para poder emplear la señal posteriormente en el procesamiento (codebook). El primer bloque que procesa las señales es un filtro paso bajas que permite limitar la información se va a ser procesada, el ruido ambiental de alta frecuencia es removido mediante este proceso.

El segundo bloque, modifica las características frecuenciales de las siete señales dadas. Esto es con el propósito de aumentar el nivel de las componentes de alta frecuencia de las señales de voz. Posteriormente, el ventaneo de Hamming adecua la señal en frecuencia. La detección de inicio y fin de la señal nos permite recortar la señal de manera que la información que pasará a la siguiente etapa no sea ruido o muestras parásitas.

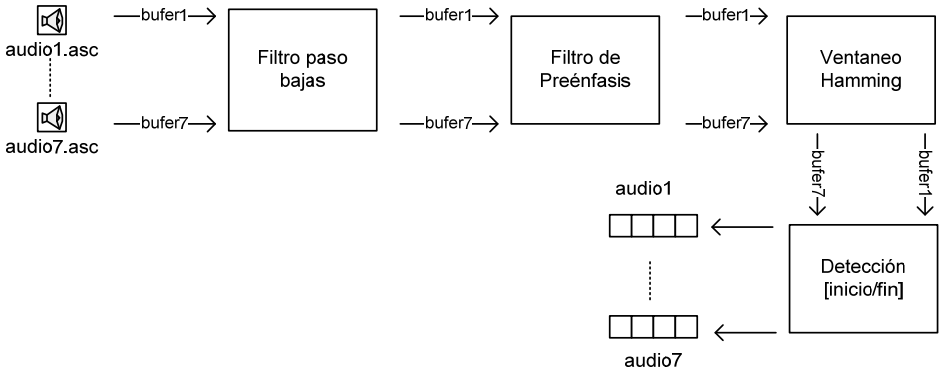


Figura 5.4 Procesamiento de las señales o fase de análisis

El siguiente conjunto de procesos es el encargado de obtener los centroides del código que está siendo entrenado. Cada audio es segmentado y a partir de esto obtenemos palabras que a su vez son procesadas para caracterizarlas con sus LPC's. Posteriormente el algoritmo de K-Medias procesa las cuatro palabras diferentes para obtener sus respectivos libros de centroides.

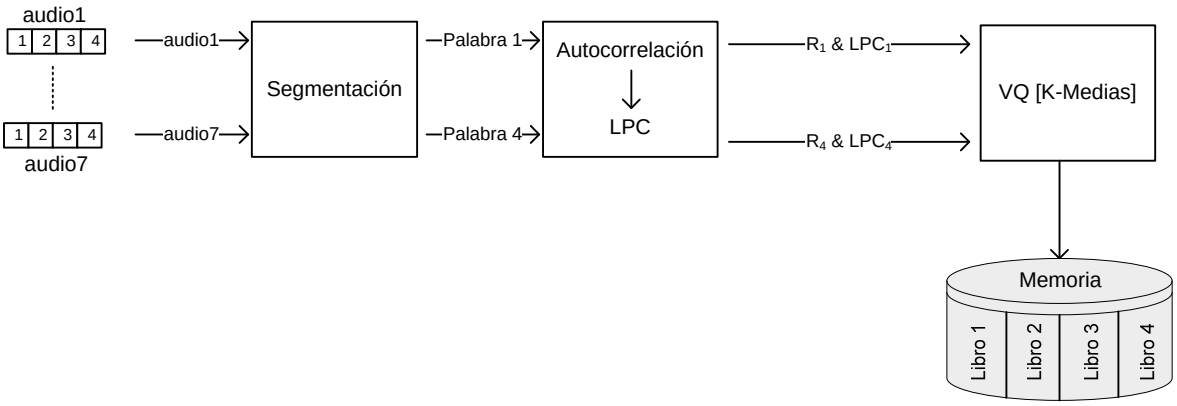


Figura 5.5 Obtención de centroides y algoritmo K-Medias

5.3 Reconocimiento

El proceso de reconocimiento se muestra en la figura 5.6, este empieza con grabar la señal de audio, reproducirla para que el usuario este conciente de lo que se está mandando al servidor. Al mismo tiempo que se reproduce la señal, esta es enviada al servidor esto mediante tareas paralelas (hilos) en Java Micro Edition.

En el servidor se regenera el archivo de audio y se convierte al formato *.wav, una vez almacenado el audio se dispara el reconocedor de voz y el servidor envía el nombre del comando reconocido al celular si se va a disparar un proceso o bien imprime el contenido relacionado con el comando reconocido. Si el comando reconocido es el correcto el usuario envía una confirmación para disparar el proceso asignado a dicho comando de lo contrario podemos regresar e intentar un nuevo reconocimiento.

5.3.1 Reconocedor

Una vez que la señal de audio ha sido restaurada por el servidor, entra a un proceso de caracterización mediante sus LPC's y autocorrelación, posteriormente la señal entra al proceso de comparación vectorial que utiliza todos los centroides almacenados durante el entrenamiento. Una vez que el algoritmo termina, el servidor realiza una acción con base en el comando recocido. Esto se observa en la Figura 5.7

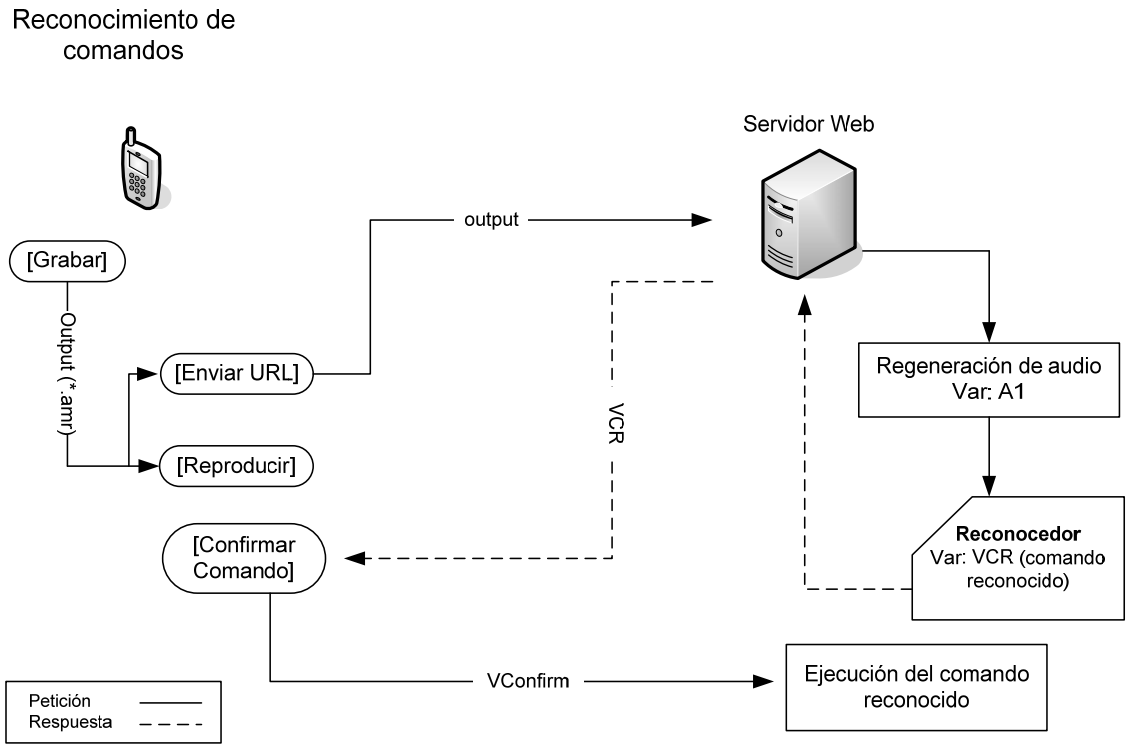


Figura 5.6 Diagrama de reconocimiento

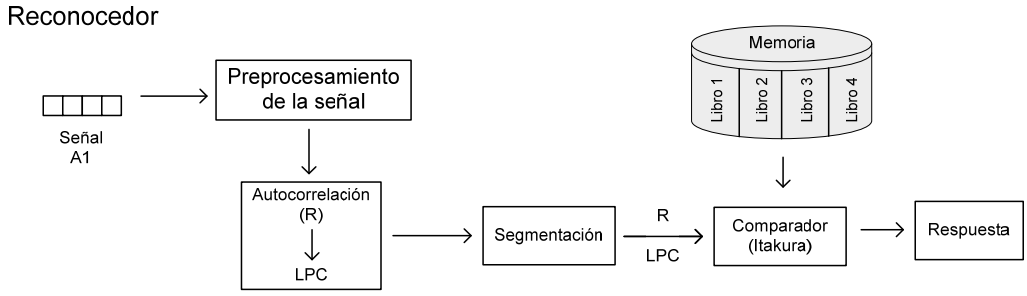


Figura 5.7 Proceso de reconocimiento

5.4 Ejecutar

El proceso comienza cuando el usuario selecciona la opción Ejecutar, en ese momento se envía una petición al servidor y este contesta con una lista de comandos almacenados en la base de datos, la cual es mostrada en el celular para que el usuario seleccione alguno de estos comandos. Por último se ejecute el proceso asociado a dicho comando.

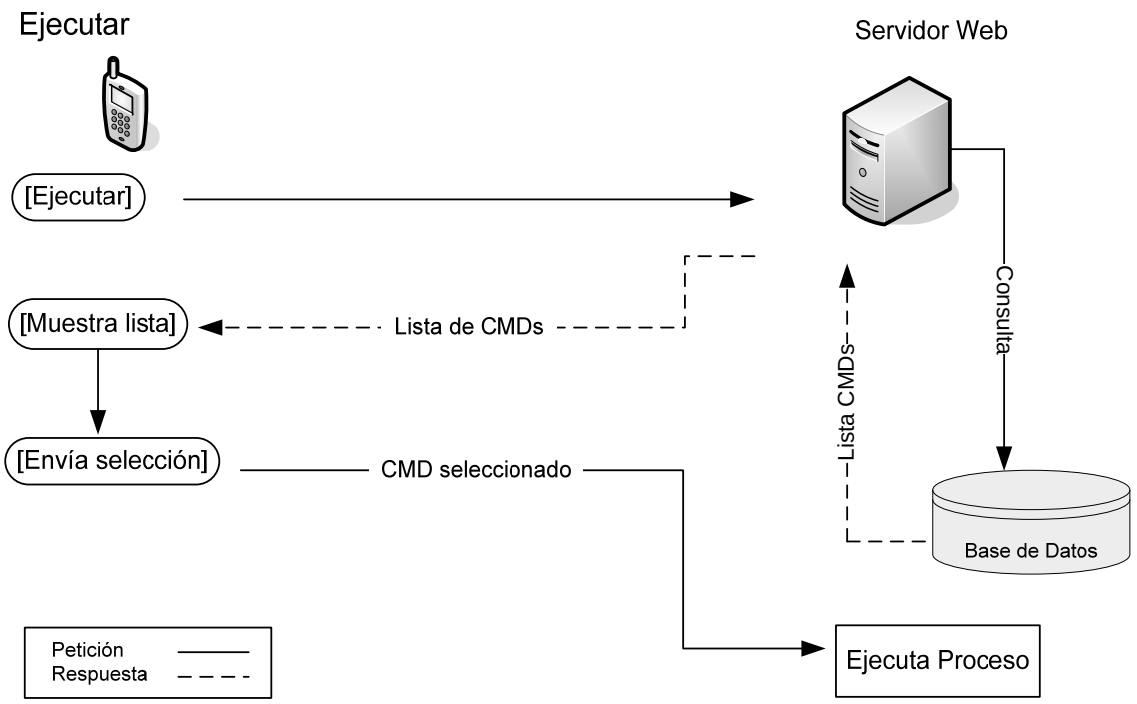


Figura 5.8 Diagrama de Ejecución

5.5 Interfaz de usuario en el servidor

Del lado del servidor se cuenta con una serie de páginas Web que sirven como interfaz de usuario en las cuales podemos crear nuevos usuarios del sistema o ver los comandos almacenados para asignarles algún proceso. Debido a que existen 2 reconocedores de voz uno en Matlab y el otro en PHP tuvimos que realizar ligeras variaciones a la interfaz las cuales se describen continuación.

Si deseamos ingresar al sistema realizado en Matlab tecleamos la URL http://127.0.0.1/reconoce/ en el navegador web y para ingresar al sistema realizado en PHP tecleamos http://127.0.0.1/reconoce_PHP/ para los dos casos la pantalla de inicio se muestra en la figura 5.9:

Figura 5.9 Pantalla de inicio

En la imagen anterior (Figura 5.9) tenemos la opción de crear un registro si es que somos un nuevo usuario o ingresar el login y password para ver los comandos almacenados, para el primer caso demos clic en crear registro y veremos en pantalla la figura 5.10:

Figura 5.10 Creación de registro

Ingresamos el login, password y signo zodiacal los cuales se guardaran en la base de datos.

Una vez creado el registro y entrenado por medio del celular algunos comandos podemos ingresar al sistema para asignarle algún proceso a dichos comandos.

Figura 5.11 Sesión del usuario creado

Para ver los comandos almacenados hacemos clic en el botón “Ver Comandos Guardados” y si estamos en la versión de Matlab veremos la siguiente figura:

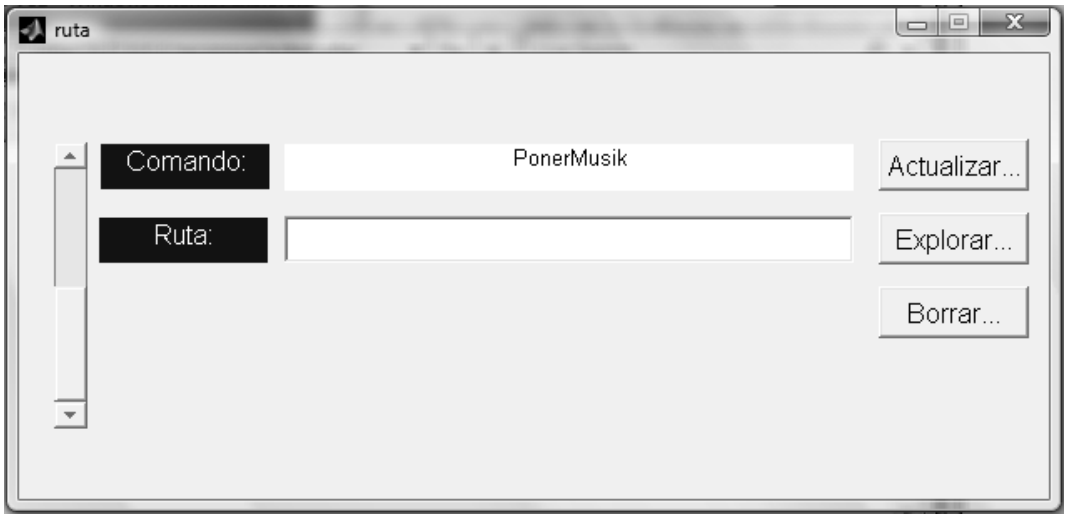


Figura 5.12 Asignación del proceso a ejecutar en los comandos almacenados

Para este ejemplo, previamente se entreno el comando PonerMusik, en la interfaz tenemos tres opciones *[Actualizar…]*, *[Explorar…]* y *[Borrar…].* Si damos clic en *[Explorar…]* veremos en pantalla la figura 5.13., en la cual podremos elegir un archivo al que estará ligado nuestro comando PonerMusik. En este caso puede ser un archivo de música o una lista de reproducción, sin embargo también puede tratarse de un archivo ejecutable *.exe, un archivo de sistema de Windows *.bat o cualquier otro archivo, además de poder ingresar direcciones de Internet como www.google.com.mx.

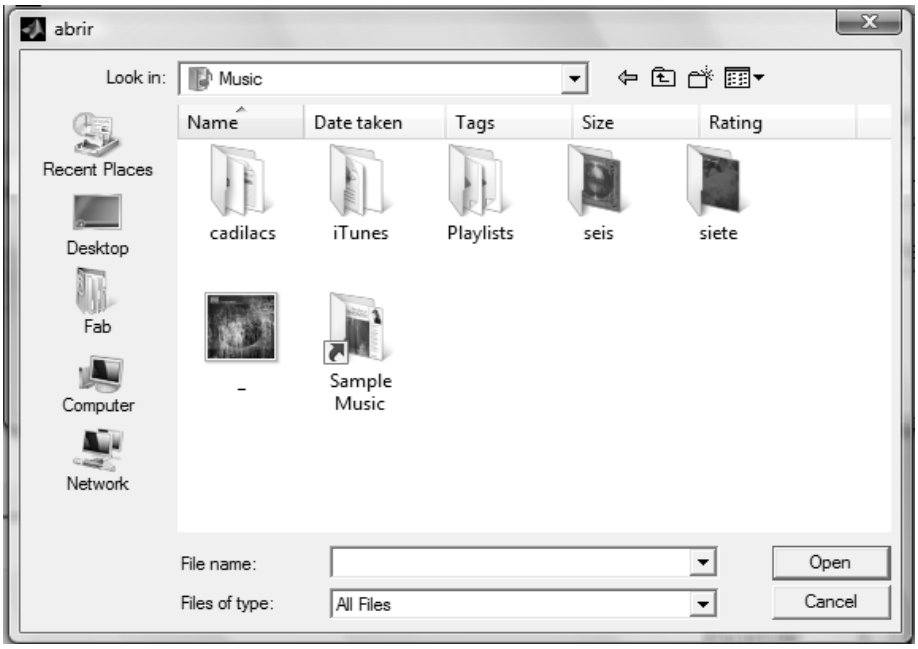


Figura 5.13 Asignación de archivo a ejecutarse.

Una vez seleccionado el archivo aparecerá la ruta en el segundo campo de la interfaz, e inmediatamente después damos clic en Actualizar… para grabar la ruta en la base de datos.

Por último tenemos la opción *[Borrar…]* con la cual eliminamos algún comando de la base de datos así como sus correspondientes centroides.

Si entramos en la sesión de PHP la interfaz se muestra en la figura 5.14:

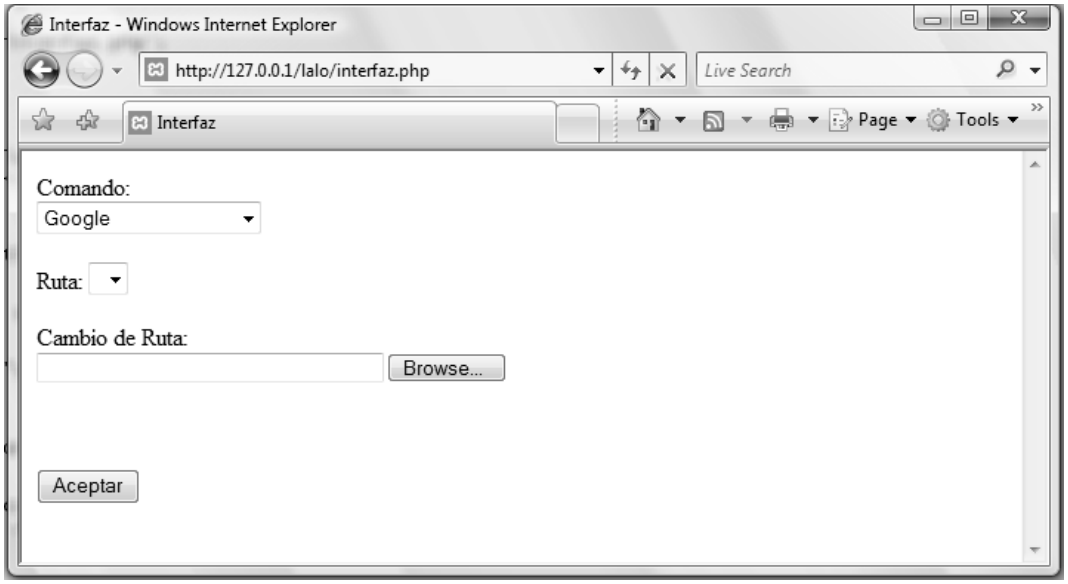


Figura 5.14 Asignación de comandos en PHP

El botón *[Browse…]* sirve para buscar la ruta a algún archivo y Aceptar para grabar dicha ruta en la base de datos.

Una vez guardadas las rutas a los archivos cada vez que envié una confirmación del celular al servidor para ejecutar algún proceso ligado a un comando serán estos archivos los que se ejecutaran.

6 PRUEBAS Y CONCLUSIONES

6.1 Pruebas del sistema.

Una vez que el prototipo del sistema de comunicación estuvo terminado, cada desarrollador realizó pruebas sobre las dos plataformas de desarrollo. Las pruebas consistieron en entrenar diez palabras con cada una de las plataformas (Matlab y PHP) y posteriormente realizar diez pruebas de reconocimiento sobre cada palabra en cada plataforma.

Los resultados de estas pruebas se muestran a continuación.

Grado de Reconocimiento							
	Excelente		Bueno		Regular		Malo

Tabla 6.1 Código de colores de la calidad del reconocedor

		1	2	3	4	5	6	7	8	9	10	Promedio de %
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	8		1							1	
2	Apaga PC		7					1				
3	Abre Word		1	9								
4	Poner Música				9							
5	Google					9						
6	Prender Luz				1		10		1			
7	Apagar Luz	1	1					9				
8	Marcador								9			
9	Horóscopo					1				10		
10	Noticias	1	1								9	
	%acierto	80	70	90	90	90	100	90	90	100	90	80
	%error	20	30	10	10	10	0	10	10	0	10	10

Tabla 6.2 Usuario Daniel – Pruebas sobre la plataforma MATLAB

		1	2	3	4	5	6	7	8	9	10	Promedio de %
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	10				1						
2	Apaga PC		10					1	1			
3	Abre Word			9					1			
4	Poner Música				9							
5	Google					9						
6	Prender Luz						10					
7	Apagar Luz							9				
8	Marcador			1					8			
9	Horóscopo									10		
10	Noticias				1						10	
	%acierto	100	100	90	90	90	100	90	80	100	100	94
	%error	0	0	10	10	10	0	10	20	0	0	6

Tabla 6.3 Usuario Daniel – Pruebas sobre la plataforma PHP

		1	2	3	4	5	6	7	8	9	10	%Promedio de porcentajes
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	10										
2	Apaga PC		9					1				
3	Abre Word			10		1			2			
4	Poner Música				7		1					
5	Google					9						
6	Prender Luz						9					
7	Apagar Luz		1		3			8		1		
8	Marcador							1	8			
9	Horóscopo									9		
10	Noticias										10	
	%acierto	100	90	100	70	90	90	80	80	90	100	89
	%error	0	10	0	30	10	10	20	20	10	0	11

Tabla 6.4 Usuario Fabián – Pruebas sobre la plataforma MATLAB

		1	2	3	4	5	6	7	8	9	10	%Promedio de porcentajes
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	8					1				1	
2	Apaga PC	1	9						1	1		
3	Abre Word			9								
4	Poner Música				8		1	1				
5	Google					10						
6	Prender Luz	1			1		8					
7	Apagar Luz			1	1			9				
8	Marcador		1						9			
9	Horóscopo									9		
10	Noticias										9	
	%acierto	80	90	90	80	100	80	90	90	90	90	88
	%error	20	10	10	20	0	20	10	10	10	10	12

Tabla 6.5 Usuario Fabián – Pruebas sobre la plataforma PHP

		1	2	3	4	5	6	7	8	9	10	Promedios de %
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	10										
2	Apaga PC		8					2				
3	Abre Word			7								
4	Poner Música				9	1		1		1	1	
5	Google					9						
6	Prender Luz				1		10					
7	Apagar Luz		2	3				7				
8	Marcador								10			
9	Horóscopo									9		
10	Noticias										9	
	%acierto	100	80	70	90	90	100	70	100	90	90	88
	%error	0	20	30	10	10	0	30	0	10	10	12

Tabla 6.6 Usuario Lizbeth – Pruebas sobre la plataforma MATLAB

		1	2	3	4	5	6	7	8	9	10	Promedio de %
		Mi PC	Apaga PC	Abre Word	Poner Música	Google	Prender Luz	Apagar Luz	Marcador	Horóscopo	Noticias	
1	Mi PC	10										
2	Apaga PC		9									
3	Abre Word			9			1			1		
4	Poner Música				9		1				2	
5	Google					10	1					
6	Prender Luz			1	1		7					
7	Apagar Luz		1					10				
8	Marcador								9			
9	Horóscopo								1	9		
10	Noticias										8	
	%acierto	100	90	90	90	100	70	100	90	90	80	90
	%error	0	10	10	10	0	30	0	10	10	20	10

Tabla 6.7 Usuario Lizbeth – Pruebas sobre la plataforma PHP

El análisis de estos resultados nos indica la exactitud con la que trabaja el sistema sobre las plataformas usadas. El desempeño general del sistema se puede observar en el siguiente gráfico.

	MATLAB		PHP	
Usuario	acierto	error	acierto	error
Liz	88	12	90	10
Daniel	89	11	94	6
Fabián	89	11	88	12
%Promedio	88.67	11.33	90.67	9.33

Tabla 6.8 Resultados porcentuales de la efectividad del sistema

El grado de efectividad del sistema desarrollado en PHP es mayor que el que presentó Matlab en un 2%. Por otro lado la velocidad de respuesta de Matlab (en promedio 3 segundos) es mayor que la de PHP. Para mantener un registro de toda actividad realizada en el sistema se implemento un algoritmo que registra el tiempo de procesamiento de cada tarea. La información recopilada por este monitor de sistema se presenta de la siguiente forma:

```
September 3, 2008, 5:36 pm || funcion: reconoce | t: 3.69994997978| p: 8 | k: 12
September 3, 2008, 5:37 pm || funcion: reconoce | t: 5.14795804024| p: 8 | k: 12
September 3, 2008, 5:38 pm || funcion: reconoce | t: 4.01275610924| p: 8 | k: 12
September 3, 2008, 5:38 pm || funcion: reconoce | t: 4.47814011574| p: 8 | k: 12
September 3, 2008, 5:39 pm || funcion: reconoce | t: 5.59799385071| p: 8 | k: 12
September 3, 2008, 5:39 pm || funcion: reconoce | t: 4.6143450737| p: 8 | k: 12
September 3, 2008, 5:40 pm || funcion: reconoce | t: 3.71637296677| p: 8 | k: 12
September 3, 2008, 5:40 pm || funcion: reconoce | t: 3.79743504524| p: 8 | k: 12
September 3, 2008, 5:41 pm || funcion: reconoce | t: 4.10359096527| p: 8 | k: 12
```

Figura 6.1 Registro de actividad y tiempo de procesamiento

El monitor registra y guarda en la carpeta de configuración de cada usuario un archivo que registra la fecha en que se realizó la tarea, la función que se realizó (entrenar o reconocer), el tiempo que la función tardó en regresar un resultado y finalmente los parámetros necesarios para obtener y operar con los LPC de las señales de voz.

El sistema está pensado para usarse en aplicaciones de control de procesos o de entrega de contenidos vía Internet, por lo que se cuenta con un alto grado de confianza en el procesamiento de la voz a nivel Servidor-PHP.

Los resultados dependieron de un gran número de variables, el ambiente o entorno donde se realizaron las pruebas de reconocimiento y donde se realizó el entrenamiento, esto implica que el ruido ambiental, la dicción del usuario, así como la manera de dar la instrucción, pueden ser variables que afecten el resultado de los procesos de entrenamiento y reconocimiento.

6.2 Propuestas de mejora

1. MEJORAR LA OPERACIÓN EN EL SERVIDOR. Para esto es necesario cambiar la programación del sistema reconocedor-entrenador a un lenguaje de programación más robusto y rápido, como C++ o Java que pueda ligarse al servidor Apache y que tenga una comunicación con PHP para generar las respuestas que se reflejarán en el móvil. Otra mejora indispensable es el manejo de señales de voz nativas de un dispositivo móvil, el procesamiento directo sobre ellas sin ningún tipo de conversión, nos permitiría reducir los tiempos de procesamiento y el número de tareas en el servidor.

2. MEJORAR EL RECONOCEDOR DE VOZ. Existen métodos de caracterización de voz más robustos y exactos que los LPC, como son los modelos ocultas de Markov (HMM) entre otros. La aplicación de uno de estos métodos puede incrementar el porcentaje de acierto en el sistema. Otra mejora que incrementaría el reconocimiento es la detección y eliminación de ruido, lo cual implica el estudio de este tema fundamental en procesamiento de señales. Una tercera mejora en el reconocedor consistiría en colocar bloques intermedios en los procesos de reconocimiento y entrenamiento que permitan trabajar con los silencios de las palabras ligadas.

3. AUMENTAR LA FUNCIONALIDAD DEL DISPOSITIVO MOVIL. El desarrollo de nuevos sistemas operativos para teléfonos móviles y el incremento en las funciones y servicios que estos dispositivos nos ofrecen, nos permite pensar en el desarrollo de un sistema reconocedor – entrenador que opere directamente en el móvil. La interacción con el servidor nos permitirá disparar procesos externos o transferir contenidos a partir del comando reconocido en el mismo móvil. Se puede modificar todo el sistema para incrementar la efectividad del proceso de entrenamiento siempre y cuando la cantidad de señales de voz que se utilicen para entrenar sea mayor, esto depende de las características del móvil.

6.3 Conclusiones

Los objetivos de esta investigación fueron en su mayoría alcanzados ya que se logró obtener un prototipo funcional del sistema de comunicación, el cual se encarga de enlazar a un dispositivo móvil con un servidor y ejecutar procesos por medio de comandos de voz.

El trabajo se inicio en Matlab, un software para desarrollo científico y posteriormente se migro a PHP, un lenguaje de desarrollo para aplicaciones web, que tiene entre sus características más importantes el ser multiplataforma. La posibilidad de crear un sistema de reconocimiento de voz como aplicación web nos permitirá generar servicios a nivel internet capaces de satisfacer nuevas necesidades.

El trabajo realizado en este proyecto estuvo definido por las características de los dispositivos móviles que utilizamos y por las plataformas empleadas. La aplicación desarrollada para el dispositivo móvil fue probada en los celulares modelos Z610i y W610i de la marca Sony Ericsson y puede ser ejecutada en dispositivos que cuenten con la capa de configuración CLDC 1.1 y capa de perfil MIDP 2.0. La diferencia entre la eficiencia de los dos prototipos diseñados depende de los propósitos generales de cada plataforma empleada.

Existen muchos parámetros que afectan el reconocimiento de patrones de voz, el ruido, los silencios entre palabras, el hablante, la compresión de la voz en el móvil e incluso las características de los transductores (micrófonos). El poder crear sistemas que contemplen todos estos factores depende de un desarrollo multidisciplinario donde varias ramas de la ingeniería deben de trabajar para obtener buenos resultados.

Los alcances del sistema aún son limitados, la capacidad de los dispositivos móviles aumenta y debemos adecuar el software del móvil para explotar las nuevas tecnologías, sin embargo, el sistema propuesto presenta la ventaja de que con características mínimas, es decir, dispositivos que sean capaces de grabar la voz y cuenten con una conexión a red. Las mejoras del sistema dependen de aprender más sobre cómo funcionan los dispositivos móviles, es decir, conocer a profundidad el software y hardware que tenemos en nuestras manos.

Entre los problemas que se tuvieron para el desarrollo de este prototipo se encuentran las conversiones de formato de la señal, lo cual afecta en el tiempo de procesamiento.

Al desarrollar este proyecto hemos adquirido conocimientos de nuevas tecnologías pero sobre todo hemos entrado en el estudio de un área de la ingeniería muy amplia y que aun requiere mucho trabajo e investigación para su desarrollo y crecimiento. Los sistemas de procesamiento de voz cada vez se hacen más presentes en la vida cotidiana y es indispensable saber que fundamentos se encuentran detrás de ellos si queremos desarrollar esa tecnología.

Bibliografía

- [1] G. Fant. Acoustic theory of speech production. The Hague,1970.

- [2] J. L. Flanagan. Speech Analysis, Synthesis and Perception. Springer-Verlag, 2d edition, 1972.

- [3]Rabiner,L. Fundamentals of Speech Recognition. New Jersey: Prentice Hall, 1993.

- [4] D.Kornhauser. Desarrollo de un reconocedor de voz público de palabras aisladas, 1999.

- [5] Agustín Froufe: "J2ME: Manual de usuario y tutorial". Ra-Ma.

- [6] Sergio Gálvez y Lucas Ortega: "Java a tope: J2ME (Java 2 Micro Edition)". Edición electrónica.

- [7] JSR 135: Mobile Media API
  http://jcp.org/en/jsr/detail?id=135

- [8] Forum Nokia. AMR Format
  http://wiki.forum.nokia.com/index.php/AMR_format

- [9] SourceForge.net. AMR2Wav Converter
  http://sourceforge.net/projects/amr2wav

- [10] MySQL. Sun Microsystems
  http://www.mysql.com/

- [11] PHP
  http://www.php.net/

- [12] Taller PHP
  http://www.programacion.net/php/articulo/porquephp

- [13] PHP
  http://es.wikipedia.org/wiki/.php

- [14] The Conceptual Architecture of the Apache Web Server
http://www.cs.ucsb.edu/~tve/cs290i-sp01/papers/Concept_Apache_Arch.htm

- [15] Wireless Network First-Step. Geier Jim. Cisco Press. USA 2005.

- [16]Bluetooth Demystified. Muller Nathan. McGraw-Hill TELECOM. USA 2000.

Apéndice A

Manual de Usuario

1. Instalación de Eclipse y del Plugin Wireless Toolkit 2.5.1 de Sun Microsystems para programación en Java Micro Edition (J2ME).

2. En la aplicaciónComandoDeVoz (desarrollada en esta tesis) **autenticamos** por medio de un **login** y **password** para identificar al usuario que está ingresando.

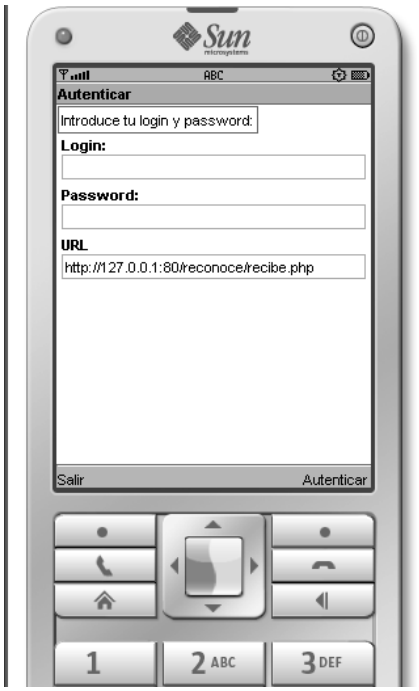


Figura A.1 Autenticación

3. **Si desea grabar:** Para llevar a cabo esta función es necesario tener al menos un comando grabado de lo contrario marcara un error. Cuando termina de grabar reproduce el audio para que el usuario verifique lo que enviara al servidor.

Figura A.2 Opción en el menú-Grabar

Inmediatamente después confirma que el comando fue grabado exitosamente, si no estamos de acuerdo con lo grabado ya sea porque no estuvimos alerta o por que el ruido a nuestro alrededor fue muy alto tenemos la opción de **Regresar** para volver a grabar sino podemos seleccionar la opción **Enviar** para mandar la señal al servidor.

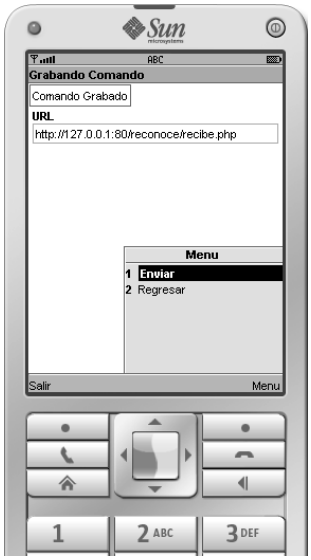


Figura A.3 Opción Enviar la señal al sevidor

Posteriormente, la aplicación espera la respuesta del servidor con el nombre del comando reconocido.

Figura A.4 Comando reconocido

Si estamos de acuerdo con el comando reconocido, es decir, que no hubo error en el reconocimiento de voz entonces pulsamos la opción de **Confirmar** la cual envía una señal de

confirmación al servidor y por lo tanto ejecutara la acción deseada que para este ejemplo se trata de direccional a la página www.google.com.mx en el servidor.

Figura A.5 Confirmación del comando reconocido

4. **Si desea entrenar:** Para entrenar un nuevo comando de voz es necesario introducir el nombre de este y la URL (Uniform Resource Locator) a la que se va a enviar, es importante resaltar que la URL que engrasamos en el proceso de autenticación se mantiene almacenada en la base de datos del celular y por lo tanto no se tiene que ingresar cada vez que se mande un comando.

Figura A.6 Registro de un nuevo comando

Si seleccionamos la opción **Continuar** y visualizaremos el mensaje **Repite 7 veces…,** seleccionamos nuevamente **Continuar** y visualizaremos los mensajes **Grabando…**, **Enviando…** y una vez enviado el comando el servidor regresa un mensaje de confirmación lo cual indica que el comando fue grabado exitosamente en el servidor.

Figura A.7 Opción continuar para seguir grabando

Seguimos seleccionando **Continuar** hasta completar las 7 repeticiones, en el séptimo comando se tarda un poco más ya que es en este donde el servidor ejecuta el entrenamiento del comando. La opción **Regresar** la utilizaremos en caso de que tuviésemos error al grabar 1 de los 7 audios, al seleccionar este el celular automáticamente vuelve a grabar el audio y lo envía. La instrucción **Inicio** nos regresa al menú principal

Figura A.8 Opción continuar para llevar a cabo el entrenamiento

5. **Si desea ejecutar:** En esta opción podremos elegir un comando de una lista, la cual es desplegada al usuario después de consultar en la base de datos del servidor. Estos comandos deben de estar previamente guardados en la base de datos.

Figura A.9 Lista de comandos a ejecutar

Una vez seleccionado este comando se ejecutara el proceso correspondiente en el servidor y si todo fue bien veremos la siguiente pantalla:

Figura A.10 Confirmación de ejecución

El mensaje estará por unos segundos y después regresará la lista de comandos por si se desea seleccionar algún otro comando.

Apéndice B

Interfaz Teléfono Celular – Servidor (Bluetooth)

Bluetooth es un estándar empleado en enlaces de radio de corto alcance, destinado para reemplazar el cableado existente entre dispositivos electrónicos móviles como teléfonos celulares, Asistentes Personales Digitales (o sus siglas en Inglés PDA) o dispositivos fijos como computadoras, teléfonos fijos y muchos otros dispositivos de una manera segura.

Con la tecnología inalámbrica Bluetooth pueden realizarse comunicaciones punto a punto y punto-multipunto. Cada dispositivo Bluetooth estar equipado con un microchip que transmite y recibe en una banda de frecuencias que esta disponible en todo el mundo (con algunas variaciones de ancho de banda en diferentes países). Una de las características de la tecnología Bluetooth es la capacidad de gestión simultánea de señales de voz (se tienen disponibles tres canales de voz) y datos. Cada dispositivo tiene una dirección única de 48 bits basado en el estándar IEEE 802.

La tecnología Bluetooth opera en una banda de frecuencia industrial, científica y médica (ISM) que no requiere licencia y que se encuadra, concretamente, entre 2.4 y 2.485 GHz. Utiliza una señal bidireccional en un espectro ensanchado por salto de frecuencia a una velocidad nominal de 1600 saltos/segundo.

El equipo de transmisión se clasifica en 3 grupos según el nivel de potencia de emisión y el alcance depende de la clase del dispositivo:

- Los radios de clase 3 suelen tener un alcance de entre uno y tres metros.
- Las radios de clase 2 son habituales de los dispositivos portátiles y tienen un alcance de diez metros.
- Las radios de clase 1 se utilizan principalmente en el sector industrial y logran un alcance de cien metros.

El equipo receptor debe poseer una sensibilidad de al menos -70 dBm, y la tasa de error admisible debe ser menor ó igual a 0,1 %. La Potencia de transmisión que se usa como especificación es de 1 mW para un alcance de 10 m, 100 mW para un alcance de hasta 100 m. Las radios más utilizadas son las de clase 2, con una potencia de 2,5 mW. La tecnología Bluetooth se diseñó para minimizar el consumo de energía. Para ello, la especificación cambia las radios al modo de ahorro de energía cuando no están activas.

La velocidad de transmisión del estándar Bluetooth se define en la siguiente tabla.

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
UWB Bluetooth	53 - 480 Mbit/s

Tabla B.1 Velocidad del estándar Bluetooth

Protocolos Bluetooth

De manera análoga al sistema OSI, bluetooth aproxima su arquitectura de protocolos como niveles o capas. De esta estructura vemos que la base de la arquitectura de bluetooth permite desarrollar aplicaciones que interactúan con todas las partes de la jerarquía. Esto nos lleva a la operabilidad entre aplicaciones de dispositivos remotos que corren bajo las mismas reglas de arquitectura.

No todas las aplicaciones utilizan todos los protocolos del conjunto de protocolos de bluetooth (Bluetooth Protocol Stack), de hecho pueden correr sobre uno o mas bloques verticales del conjunto.

Diferentes aplicaciones pueden operar bajo distintos conjuntos de protocolos; sin embargo, todos ellos tienen un enlace de datos y una capa física Bluetooth común.

Los protocolos pueden ser divididos en cuatro capas:

- Protocolos Bluetooth Centrales (Bluetooth Core Protocols: BaseBand, LMP, L2CAP, SDP).
- Protocolos de Reemplazo de Cable (Cable Replacement Protocols: RFCOMM).
- Protocolos de control de Telefonía (Telephony Control Protocols: TCS Binary, AT-Commands).
- Protocolos Adaptados (Adapted Protocols: PPP, UDP/TCP/IP, OBEX, WAP, vCard, vCal, IrMC, WAE).

Apéndice C

Protocolo TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) es un conjunto de protocolos desarrollados para permitir que cooperen las computadoras a fin de compartir recursos a través de una red. Cualquier dispositivo que se desea interconectar a Internet necesita ejecutar el set de protocolos TCP/IP.

El modelo TCP/IP tiene cuatro capas: la capa de aplicación, la capa de transporte, la capa de Internet y la capa de acceso a red.

La ***Capa de Aplicación*** manipula protocolos de alto nivel y temas de representación, codificación y control del diálogo. TCP/IP cuenta con un conjunto de protocolos para dar soporte a la transferencia de archivos, al correo electrónico y la conexión remota.

- Protocolo de Transferencia de Hipertexto (HTTP). Define el formato y el modo de transmisión de los mensajes, así como las acciones que deben llevar a cabo los servidores y navegadores web en respuesta a distintos comandos.
- Protocolo Trivial de Transferencia de Archivos (TFTP). Es utilizado para transferir archivos de configuración entre sistemas que lo soportan.
- Protocolo de Transferencia de Archivos (FTP). Es un servicio orientado a conexión que utiliza TCP para transferir archivos entre sistemas que lo soportan.
- Sistema de Archivo de Red (NFS). Conjunto de protocolos desarrollados por SUN Microsystems que permite el acceso remoto a archivos a través de una red.
- Protocolo Simple de Transferencia de Correo (SMTP). Protocolo que dirige la transmisión de correo electrónico sobre redes computacionales. Soporta intercambio de texto exclusivamente.
- Emulación de Terminal (TELNET). Proporciona la capacidad de acceder remotamente a otra computadora. Permite al usuario conectarse a un *host* local y ejecutar comandos.
- Protocolo Simple de Administración de Redes (SNMP). Protocolo que permite supervisar y controlar los dispositivos de una red, administrar configuraciones, recopilar estadísticas, rendimiento y seguridad.
- Sistema de Denominación de Dominio (DNS). Sistema utilizado en Internet para convertir los nombres de Internet y sus nodos de red anunciados públicamente en direcciones IP.

La ***Capa de Transporte*** proporciona servicios de transporte desde un host de origen a un host de destino. Constituye una conexión lógica entre los extremos de la red: host emisor y el host receptor. Los protocolos de transporte TCP y UDP segmentan y reensamblan los datos que las aplicaciones de las capas superiores envían.

La ***Capa de Internet*** tiene como finalidad enviar paquetes desde un dispositivo utilizando el protocolo correcto que opera en esta capa. En esta capa se determina la mejor ruta y la conmutación de paquetes.

- IP. Proporciona el enrutamiento de paquetes sin conexión. Su propósito es llevar los paquetes a su destino.
- Protocolo de Mensajes de Control de Internet (ICMP). Proporciona capacidades de control y mensajería.
- Protocolo de Resolución de Direcciones (ARP) determina las direcciones de la capa de enlace de datos (direcciones MAC) para las direcciones IP conocidas.
- Protocolo de Resolución Inversa de Direcciones (RARP). Determina direcciones IP cuando se conocen las direcciones de la capa de enlace de datos.

La ***Capa de Acceso a Red*** se ocupa de los temas que un paquete requiere para crear un enlace físico con el medio de una red. Incluye los detalles de las tecnologías LAN y WAN y los detalles contenidos en la capa física y de enlace de datos del modelo OSI.

Métodos GET Y POST

- **GET:** Es un método definido en HTTP para obtener un archivo y envía una pequeña cantidad de datos usando la URL. La longitud de GET está delimitada por el espacio libre de los buffers de entrada.

- **POST:** Se utiliza para enviar una cantidad mayor de información al servidor, mientras que con el método GET se usa la URL con POST se envía por la entrada estándar STDIO. Este método invoca procesos los cuales generan datos que son devueltos como respuesta a una petición.