PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

SEDE AMBATO

ESCUELA DE INGENIERÍA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA A LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

"DESARROLLO DE UN SISTEMA DE MULTIDIFUSIÓN DE MENSAJES DE VOZ EN LA ESCUELA DE SISTEMAS DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO"

RINA KATHERINE SÁNCHEZ REINOSO

FRANCISCO XAVIER MEDINA BRAVO

DIRECTOR DE LA DISERTACIÓN:

ING. SANTIAGO ACURIO

AMBATO, 2004

Miriam Nieri de Mesa

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

SEDE AMBATO

ESCUELA DE INGENIERÍA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA A LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

"DESARROLLO DE UN SISTEMA DE MULTIDIFUSIÓN DE MENSAJES DE VOZ EN LA ESCUELA DE SISTEMAS DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO"

RINA KATHERINE SÁNCHEZ REINOSO

FRANCISCO XAVIER MEDINA BRAVO

DIRECTOR DE LA DISERTACIÓN:

ING. SANTIAGO ACURIO

AMBATO, 2004

DEDICATORIA

Quiero dedicar este trabajo, fruto de mi esfuerzo y capacidad, primeramente a Dios, quien me ha guiado y bendecido en mis estudios, por un camino de constante lucha, sacrificio y superación.

A mis padres quienes con esfuerzo y sacrificio, han impulsado mi superación día a día, brindándome apoyo y seguridad para llegar a ser una profesional.

A mi esposo quien con su apoyo y ejemplo de lucha constante me ha enseñado que en el camino de la vida hay caídas en las que hay que levantarse y seguir adelante. Y la culminación de mis años de estudio y de este trabajo es un ejemplo de ello.

Y a mis hijos porque ellos han sido mi empuje y la fuerza para seguir adelante en la vida.

Gracias a todos.

RINA

DEDICATORIA

Todo esfuerzo, sacrificio, entrega y desvelo tiene su recompensa, es por eso que dedico este trabajo a mi madre Lic. SUSANA MEDINA MANZANO, quien con todo su sacrificio ha logrado educarme en forma total en esta prestigiosa institución. Tu esfuerzo de todos estos años es hoy tu recompensa mamita, al verle a tu primer hijo graduado y que encamina en el largo trajín de la vida.

Hubo momentos de desmayo y de frustraciones a lo largo de la realización de este trabajo, pero ahí estuvo mi esposa Rina Katherine para darme aliento y fuerza para seguir adelante, es por eso que este triunfo te lo dedico a ti mi Ñata linda.

Y en esta dedicatoria no podía faltar mis tres hijos Diego Alejandro (Permiso para irr), Rina Susana (Pachi-Pachi) y Rina Camila (Biyi-Biyi); que con sus momentos de cariño, ternura, rabietas, travesuras, me daban a entender que tengo que superarme y darles ejemplo de temple, perseverancia, constancia y principalmente enseñarles que la familia Medina Sánchez siempre llega al final y cumple sus metas.

Y todas las personas que de una u otra manera colaboraron en la realización del trabajo: Mi tía Blanca Medina Manzano, Mi prima Ing. Gloria Arcos Medina, Msc., Mi Primo Ing. Franklin Arcos Medina, Mi Padre LIc. Edmundo Poveda, Mi hermano Josué Poveda, Mi papi Dr. Juan Francisco Medina Manzano, Mis hermanos: Ing. Franklin Rolando, Anita Gabriela, Juan Daniel y John Jairo Medina Gavilanez.

Y para finalizar una dedicación muy especial al Ing. Janio Jadán, Msc. quien me ha enseñado que para triunfar en la vida hay que ser humildes y de corazón puro, y que hay hacer cada cosa en su lugar y en su tiempo. La universidad me da el titulo de Ingeniero, usted Janio me obsequia cátedras de humildad y lecciones de vida. Dios le pague al Maestro de Maestros Janio Jadán.

FRANCISCO XAVIER MEDINA BRAVO

AGRADECIMIENTO

Toda creación es un regalo de Dios, es por eso que el mejor regalo que tuvo nuestra querida ciudad es ver nacer a la Pontificia Universidad Católica del Ecuador Sede Ambato, y de ahí nuestro agradecimiento a tan prestigiosa institución, que nos acogió en sus cálidas aulas, la misma que nos brindó la oportunidad de acumular conocimientos para la realización de este trabajo de disertación.

A los directivos de la Pontificia Universidad Católica del Ecuador, que con sus consejos y apertura fueron un aliento en todo instante del desarrollo de nuestra disertación.

Al Director de la Escuela de Ingeniería de Sistemas de la PUCESA, Ing. Telmo Viteri, Msc. que gracias a sus consejos prácticos y no egoístas supo guiarnos por el camino correcto y más práctico, para poder culminar nuestro trabajo de una manera pronta, concreta y de servicio para los estudiantes. A más de todos sus consejos profesionales, supo entregarnos su amistad incondicional y sincera. Y hoy decimos Dios le pague amigo Telmo.

A nuestros maestros desde primer nivel hasta décimo, que gracias a ellos hoy somos profesionales sin prejuicios y deseosos de servir a nuestra Patria, a nuestra ciudad y a nuestra institución. Muchas Gracias: Ing. Vasily Kamenestky, Ing. Janio Jadán, Ing. Rodrigo Marcial, Ing. Natasha Bayas, Ing. Víctor Hugo Paredes, Ing. Paúl Zurita, Ing. Alexis Sánchez, Ing. Pilar Urrutia, Ing. Nadia Jaramillo, Ing. Víctor Chuncha, Lic. Miguel Hernández, Ing. Telmo Viteri, Ing. David Guevara, Ing. Wigberto Sánchez, Dra. Anita Larrea, Ec. Nelson Lascano.

Al director de nuestra Disertación, Ing. Santiago Acurio, que gracias a sus conocimientos y su forma de ver la vida futurista, entregamos un trabajo de calidad y servicial para las generaciones venideras.

Un agradecimiento muy especial a nuestros padres Mario Octavio, María Mercedes, Leoncio Edmundo y América Susana, gracias a ustedes podemos enfrentarnos a una sociedad competitiva. Dios le pague queridos papacitos.

Y un agradecimiento muy enorme para la Ing. Janio Jadán, que a lo largo de toda nuestra carrera nos supo brindar su apoyo y principalmente su amistad. Janio eres un ejemplo de maestro.

Para finalizar un agradecimiento especial a nuestro amigo Edisson Andachi, que a lo largo de toda nuestra carrera supo ofrecernos sus conocimientos sin prejuicios peor con restricciones. Muchas gracias Eddy.

Rina Katherine Sánchez Reinoso

Francisco Xavier Medina Bravo

Índice

CAPITULO I

PROYECTO DE ESTUDIO

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 DELIMITACION

Al hablar del desarrollo de un sistema de difusión se pretende interconectar parlantes en todas las aulas de la Escuela de Ingeniería de Sistemas de la Pontifica Universidad Católica del Ecuador Sede Ambato incluidos los laboratorios existentes en el momento de la instalación y las oficinas pertenecientes a la misma Escuela.

Esta interconexión permitirá que un servidor de la PUCESA, con el Sistema Operativo Windows 98 o superior utilice un software que interactúe con un dispositivo construido por los estudiantes, para seleccionar la o las aulas a las que se quiera emitir un mensaje de voz, en forma local o cliente / servidor, es decir que cualquier usuario autorizado en la red de la Escuela de Sistemas, que disponga un parlante podrá emitir un mensaje sin la necesidad de dirigirse al sitio del servidor.

1.1.2 IMPORTANCIA Y JUSTIFICACIÓN

La importancia de realizar un sistema de difusión de mensajes de voz se las expone a continuación:

- ✓ La Escuela de Ingeniería de Sistemas de la Pontificia Universidad Católica del Ecuador Sede Ambato necesita obligatoriamente estar comunicada en forma total; y que mejor manera que desde cualquier oficina con su respectiva autorización se comunique a todos los estudiantes, personal docente y administrativo, las necesidades, requerimientos, etc. que posee la escuela.
- ✓ La Escuela de Ingeniería de Sistemas de la Pontificia Universidad Católica del Ecuador Sede Ambato contará con tecnología competitiva en el ámbito local ya que sería la primera en utilizar un sistema de este tipo.
- ✓ El Sistema que se pretende realizar sería ejemplo a seguir en otras Universidades.

La justificación de realizar un sistema de difusión de mensajes se las expone a continuación:

- ✓ Al realizar un sistema de difusión la Escuela de Ingeniería de Sistemas de la Pontifica Universidad Católica del Ecuador Sede Ambato estará en capacidad de optimizar el tiempo de entrega de mensajes de voz a sus alumnos en forma individual y colectiva.
- ✓ Con la emisión de mensajes de voz se pretende ubicar a autoridades, docentes, personal administrativo o estudiantes que se encuentren en las instalaciones correspondientes a la Escuela de Sistemas del edificio de la PUCESA sector del Tropezón.

Al momento de indicar que se utilizará un servidor de la PUCESA que se encarga del monitoreo de esta estructura de cableado, estamos indicando que el computador existente en la misma deberá poseer un sistema en el cual pueda controlar el envío de todos los mensajes de voz a realizarse. Este último se pretende controlar mediante un software que posea un croquis de la distribución de aulas, laboratorios existentes en la Escuela de Sistemas de la PUCESA; el cuál nos permitirá controlar él o los sitios a los que se quiere emitir los mensajes de voz. Este software además proporcionará un acceso remoto desde cualquier otro computador con el fin de que cualquier usuario registrado en el sistema envíe un mensaje al servidor y éste lo pueda emitir a los sitios seleccionados.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

- ✓ Desarrollar un sistema de multidifusión de mensajes de voz en la Escuela de Ingeniería de Sistemas de la Pontificia Universidad Católica del Ecuador, Sede Ambato.

1.2.2 OBJETIVOS ESPECIFICOS

- ✓ Permitir que cualquier oficina de la Escuela de Sistemas de la PUCESA emita mensajes de voz mediante un micrófono del computador, a uno, varios ó todos los cursos, laboratorios, oficinas existentes en la misma de manera conjunta.
- ✓ Instalar Parlantes en cada uno de los lugares indicados en el objetivo anterior, y el cableado pertinente para llegar al computador central.
- ✓ Construir un dispositivo que se conecte al computador en donde se conecten los PLUGS de los parlantes.
- ✓ Desarrollar un software que permita la administración de envío de mensajes de voz a través del dispositivo mencionado anteriormente
- ✓ Desarrollar un software para una interfaz gráfica y amigable que permita escoger el punto de difusión de mensajes.
- ✓ Desarrollar un módulo de acceso remoto que permita a un usuario enviar mensajes desde su propio computador hacia el software del servidor
- ✓ Desarrollar un módulo de autorización de usuarios que pueda emitir el mensaje.

Ingeniería de Sistemas de la PUCESA. La definición de estas reglas, será la base de procesos mecanizados, es por lo dicho anteriormente que utilizaremos la metodología denominada ADOOSI (Análisis y diseño orientado a objetos de sistemas de información).

1.4.4 METODOLOGÍA DE DESARROLLO DE HARDWARE

Para el cumplimiento efectivo del objetivo general anteriormente planteado se desarrollará un hardware que consiste en realizar una regleta donde se conecten los PLUGS de los parlantes, en el mismo que se utilizará conocimientos de Electrónica, Acústica e Ingeniería de Sistemas.

CAPITULO II

MARCO TEÓRICO

2.1 MÉTODOS DE COMUNICACIÓN INTERNA EN LA ESCUELA DE INGENIERÍA DE SISTEMAS DE LA PONTIFICA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO

2.1.1 ANTECEDENTES

Desde tiempos remotos el hombre ha intentado comunicarse, se puede hablar que desde la era prehistórica se utilizaba signos, señas y señales para que el ser humano pueda mantener comunicación con los de su especie, posteriormente vino el habla lo cuál originó una manera de comunicación muy exacta y precisa, y todo esto dio origen a otros modos de comunicación como son la escritura, la pintura, la escultura, etc.

Desde tiempos remotos el hombre se ha enfrentado con el problema de vencer las distancias y para resolverlo empezó a utilizar sus propios medios de comunicación.

La comunicación es muy importante en la vida de los seres humanos, a lo largo de la historia de la humanidad el hombre ha creado diversos medios de comunicación, los cuales son muy variados y útiles. Con los avances tecnológicos se ha logrado contar con sistemas de comunicación más eficaces, de mayor alcance, más potentes y prácticos.

Hoy en día, los medios de comunicación constituyen una herramienta persuasiva que nos permiten mantenernos en continua comunicación con los distintos sucesos sociales, políticos y económicos tanto a escala nacional como internacional.

Los principales medios de comunicación en la actualidad son: el periódico, los libros, el telégrafo, el teléfono, la radio, la televisión e Internet.

Los satélites, además, han permitido una comunicación más amplia y eficiente especialmente en la televisión y la telefonía.

Para poder entender el carácter y función de los medios masivos de comunicación en nuestra sociedad, necesitamos conocer su historia y desarrollo. Y esta es la finalidad de este documento, pero para comenzar presentamos una visión general de la evolución que ellos han tenido:

La primera etapa de la comunicación fue probablemente la era de los signos y las señales que se desarrolló en los inicios de la prehistoria, anterior al lenguaje.

Los antropólogos opinan que el hombre prehistórico entró en la era del habla y del lenguaje alrededor de 40.000 años atrás. Para el hombre Cromagnon el lenguaje ya era de uso común. Hace 5.000 años se produjo la transformación hacia la era de la escritura, la que se constituyó en una progresiva herramienta del progreso humano. Llegar a la escritura significó pasar antes por las representaciones pictográficas que reflejaban ideas hasta la utilización de letras que significaran sonidos específicos.

Otro de los mayores logros humanos a favor de la comunicación se produjo en el siglo XV con la aparición de la imprenta de tipo móviles que reemplazó a los manuscritos. La idea fue concebida por un orfebre, Johann Gutemberg, quien después de muchas pruebas descubrió un sistema único para hacer los caracteres de imprenta.

el primer satélite artificial) abrió un nuevo panorama, pronto se contó con los primeros satélites de comunicaciones.

El cine fue inventado en 1895 en Francia, por Conisy Auguste Lumiere, y a las primeras versiones de cine "mudo", se sumaron en las décadas de 1920 y 1930 el cine sonoro, los filmes en color (popularizados luego de la Segunda Guerra Mundial), el cinema Scope y otras técnicas. Su impacto sobre la sociedad fue notable. Cuando se generalizó la televisión, se puso en duda su supervivencia.

El descubrimiento de los electrones, de las ondas electromagnéticas, de los circuitos eléctricos y electrónicos, etc., sirvieron entre finales del siglo XIX y comienzos del XX para la construcción y desarrollo de instrumentos de comunicación preferentemente audiovisuales.

El siglo XX fue, en efecto, la era de la electrónica, la era atómica, la era de las comunicaciones, etc. La introducción de nuevas tecnologías modificó la lectura, el modo de vivir y de entender la realidad. Es el cambio cultural introducido por los nuevos medios de comunicación de masas, lo que va a provocar las reacciones más dispares, desde el entusiasmo más fervoroso hasta la condena más rigurosa.

Uno de los hechos más importantes e influyentes de la historia de la humanidad en los últimos siglos ha sido el desarrollo técnico. Ese desarrollo ha abarcado todos los órdenes de la vida: la producción, la vivienda, la manera de viajar, la vida rural y urbana, la forma de hacer la guerra, la ingeniería, etc. Uno de los aspectos de ese proceso ha sido el progreso de los medios de comunicación.

Cuando el 20 de julio 1969, la primera tripulación humana llega a la Luna, el suceso fue presenciado simultáneamente en todo el planeta, por centenares de millones de personas a través de sus receptores de televisión que captaban lo que estaba ocurriendo a más de 300.000 kilómetros de distancia.

La capacidad que tenemos hoy de hacer llegar nuestros mensajes a largas distancias en forma instantánea, a través de la televisión, la radio, el teléfono, la computadora o el fax, trasmitiendo casi simultáneamente datos e informaciones, nos es tan familiar que hasta actuamos con indiferencia ante ellos.

Toda persona, hogar, oficina, edificio, ciudad, provincia o país necesita estar comunicado ya que la comunicación es la base del desarrollo de una comunidad, es por eso que el desarrollo de este trabajo de disertación pretende mantener comunicado a la Escuela de Ingeniería de Sistemas de la Pontificia Universidad Católica del Ecuador Sede Ambato.

La comunicación hoy en día es sin duda un factor muy importante en el diario convivir de los seres humanos en especial de las personas que de una u otra manera intentamos estar involucrados en el día a dia del trajinar de los avances tecnológicos. Y esta misma comunicación nos conlleva en un caminar de un sinnúmero de oportunidades y opciones: tales como el INTERNET, y/o la misma difusión de voz. La posibilidad de llevar a cabo este conjunto de tareas desde un ordenador es una ventaja excepcional, ya que la necesidad de ahorrar tiempo y dinero en los consumidores de todo el planeta es una prioridad esencial dentro del agitado y difícil mundo de hoy.

Las empresas, corporaciones y usuarios comunes de la comunicación de todo el planeta buscan día a día un servicio eficiente y confiable al momento de realizar cualquiera de las tareas señaladas anteriormente, razón por la cual muchas empresas involucradas con el desarrollo de la misma no escatiman esfuerzos al momento de crear y mejorar servicios que cumplan las necesidades de los clientes.

2.1.2 SITUACIÓN DE LA RED DE LA PUCESA

La Pontificia Universidad Católica del Ecuador, Sede Ambato cuenta actualmente con dos edificios, uno ubicado en el centro de la ciudad de Ambato y otro ubicado en el sector del Tropezón de la misma ciudad, y esta investigación está enfocada a aplicarse en la escuela de Sistemas, pero en lo posterior puede ser ampliada y aplicada a otras Escuelas y por esta razón en las figuras 1, 2, 3 y 4 se muestra diagramas de red para que sean tomadas en cuenta en la caso de aplicar esta aplicación a otras escuelas.

2.1.2.1 DIAGRAMA RED OFICINAS ESCUELA DE SISTEMAS Y DEPARTAMENTO FINANCIERO.

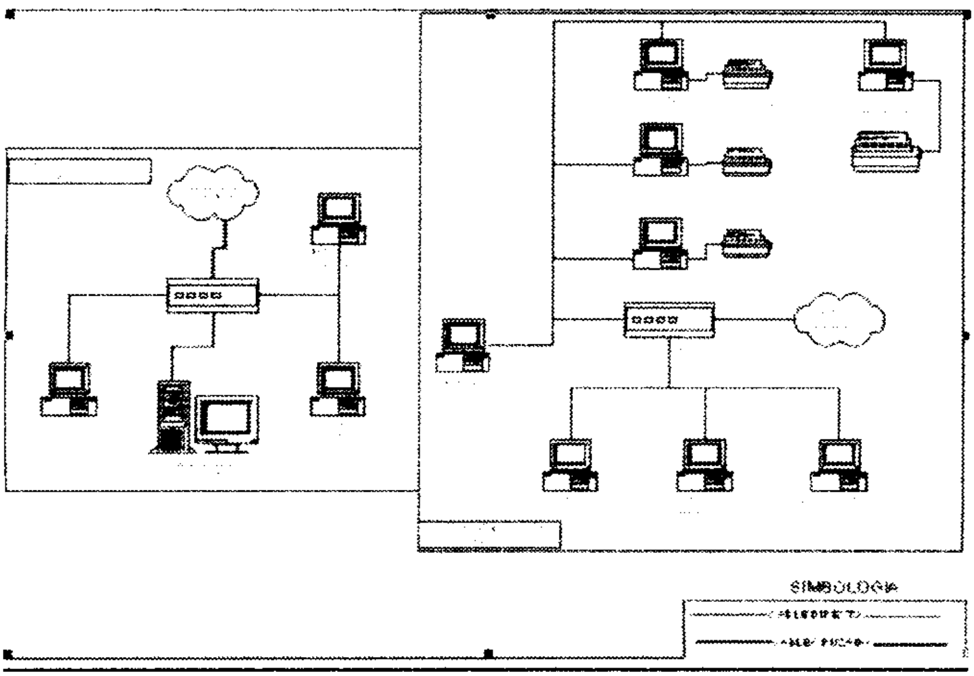


FIGURA 1: Diagrama de Red Oficinas Escuela de Sistemas y Dpto. Financiero

2.1.2.2 DIAGRAMA CENTRO DE CÓMPUTO DE LA PUCESA

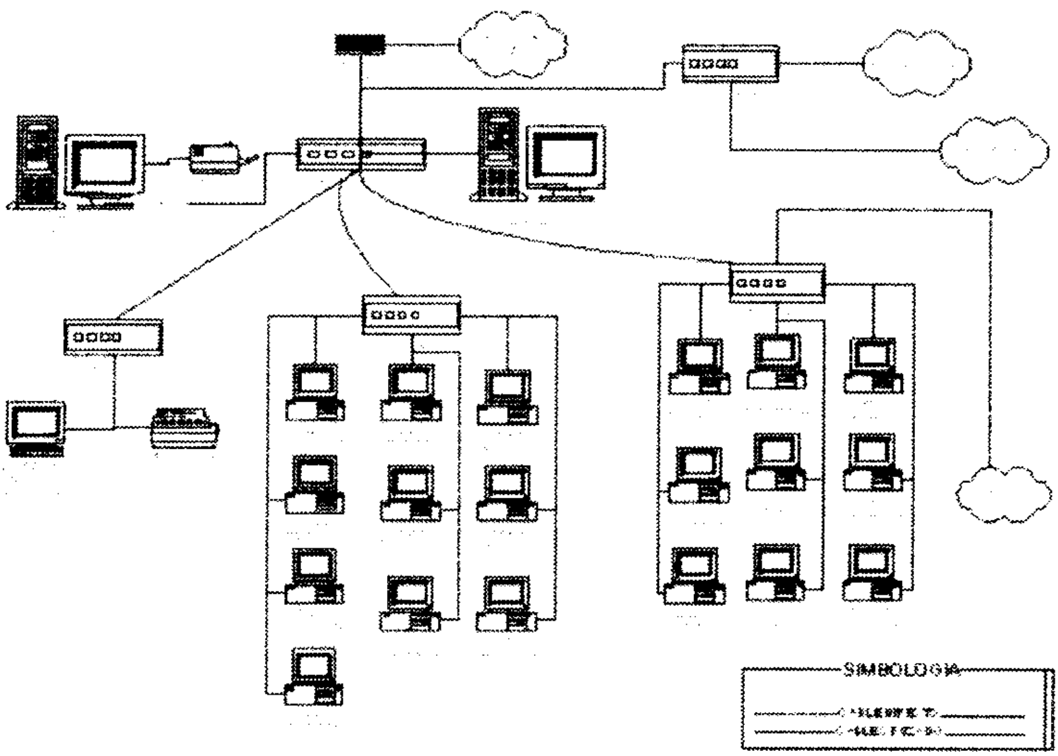


FIGURA 2: Diagrama de red Centro de Cómputo de la PUCESA

2.1.2.3 DIAGRAMA DE RED OFICINAS ESCUELA DE ADMINISTRACIÓN DE EMPRESAS

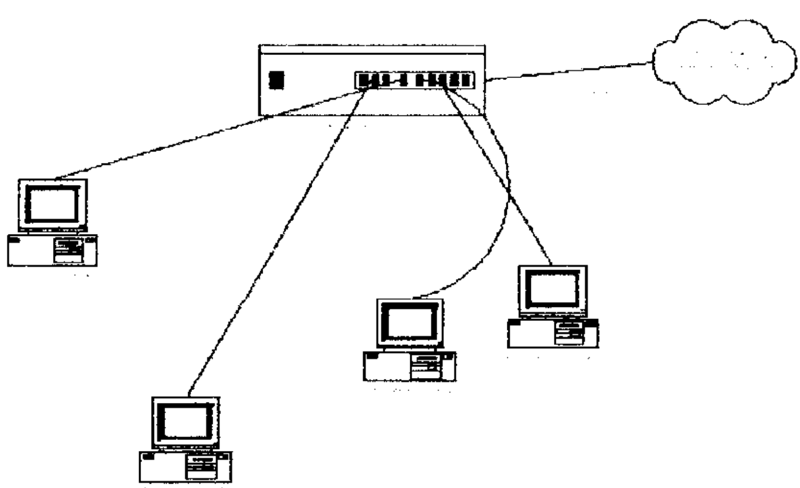


FIGURA 3: Diagrama de red Escuela de Administración de Empresas

recientes que predicen que el tráfico de voz sobre Internet puede superar al de datos en el plazo de unos pocos años. De hecho, ya el volumen de tráfico total sobre Internet supera al de voz sobre las redes telefónicas.

2.2.4. TELEFONÍA DE LARGA DISTANCIA

La VoIP es muy adecuada para dar un servicio de telefonía de larga distancia a bajo coste ya que todas las llamadas se facturan como locales. Los clientes son típicamente los *carriers* tradicionales, y una nueva categoría de ISP (Internet Service Provider, Proveedor de servicio de Internet), los ITSP (Internet Telephony Service Provider, Proveedor Telefónico de Servicio de Internet), nacida específicamente para este mercado.

En estos momentos, los grandes ahorros en cuanto a la telefonía sobre IP se realizan en las llamadas internacionales. La relativa falta de competencia en este segmento hace que los precios sean altos, y los mecanismos de compensación internacionales no favorecen la aparición de nuevos operadores con mejores precios, porque siempre tendrán que acordar cómo transportar el tráfico por las redes de los operadores existentes.

Además de la comunicación Teléfono a Teléfono, estos clientes demandan comunicaciones PC a Teléfono, servicios de Fax, enrutamiento en función del coste, tasación y contabilidad en tiempo real, etc.

2.2.4.1. COMPONENTES DE LAS REDES VOIP

Las redes de voz sobre IP suelen contener tres (o cuatro) componentes fundamentales:

- Clientes H.323, PCs multimedia conectados directamente a una red IP
- Gateways de Voz / IP
- Gatekeeper, para controlar las comunicaciones de voz sobre IP
- MCU H.323, opcional, para permitir conferencias con más de dos participantes.

2.2.4.1.1 GATEWAYS DE VOZ / IP

El Gateway de Voz/IP es el componente clave de una solución de voz sobre IP al facilitar la conversión de las llamadas telefónicas convencionales al mundo de IP.

Normalmente, tienen interfaces analógicos o digitales (PRI (Paving Roofing Industrial), PUSI) a la red telefónica, y disponen de interfaces Ethernet, Frame Relay o ATM hacia la red IP.

Gateway H.323/H.320: Básicamente, realiza la conversión entre estos dos formatos de forma que los terminales H.323 se pueden comunicar con equipos RDSI (Red Digital de Servicio Integrado) de videoconferencia, que pueden formar parte de la red corporativa o estar situados en la red pública.

Gateway H.323/RTB (Voz sobre IP). Posibilitan las comunicaciones de voz entre los terminales H.323 y los teléfonos convencionales, estén en la red corporativa o en la red pública.

2.2.4.1.2. GATEKEEPER

El Gatekeeper es un punto central de control en una red H.323, proporcionando servicios de control de llamada, traducción de direcciones y control de admisión. Además facilita el control del ancho de banda utilizado y localiza los distintos gateways y MCU's (Multimedia Control Unit, Control unido de Multimedia) cuando se necesita.

Gatekeeper H.323: Está siempre presente para controlar las llamadas en la Intranet Pública (o red corporativa). Todos los elementos de red de MMTS (terminales, Gateways, MCU) tienen que usar el Gatekeeper como punto intermedio para la señalización. De esta forma se tiene un control de los accesos, seguridad, movilidad del usuario, y tarificación si se da el caso.

2.2.4.1.3. MCU PARA H.323 Y T.120

Se utiliza cuando han de intervenir más de dos partes en una conferencia. La MCU (Multimedia Conference Unit) es responsable de controlar las sesiones y de efectuar el mezclado de los flujos de audio, datos y video.

2.2.4.1.4. LA NORMA H.323

Actualmente, las redes desplegadas para la transmisión de voz sobre IP son en su mayor parte propietarias, utilizando mecanismos de señalización, control y codificación de la voz propios de los suministradores, y con muy poca o sin ninguna interoperabilidad entre ellas. La norma H.323 de ITU viene a poner luz sobre este tema y es, a partir de ahora, prácticamente de todavía está lejano. Esta es la razón por la que la mayor parte de proveedores de voz por Internet disponen de una red dedicada para este propósito, ya que de otra manera no se puede conseguir la calidad requerida por los usuarios, sobre todo si pertenecen al mundo empresarial.

2.2.5. TARJETAS DE SONIDO

Las dos funciones principales de estas tarjetas son la generación o reproducción de sonido (representado como se indicó en el tema anterior) y la entrada o grabación del mismo. Para reproducir sonidos, las tarjetas incluyen un chip sintetizador que genera ondas musicales. Este sintetizador solía emplear la tecnología FM, que emula el sonido de instrumentos reales mediante pura programación; sin embargo, una técnica relativamente reciente ha eclipsado a la síntesis FM, y es la síntesis por tabla de ondas (WaveTable).

En WaveTable se usan grabaciones de instrumentos reales, produciéndose un gran salto en calidad de la reproducción, ya que se pasa de simular artificialmente un sonido a emitir uno real. Las tarjetas que usan esta técnica suelen incluir una memoria ROM (Read Only Memory, Memoria Solo de Lectura) donde almacenan dichos "samples"; normalmente se incluyen zócalos SIMM (Single in – Line Memory Module, Unico Modulo de Memoria en Linea) para añadir memoria a la tarjeta, de modo que se nos permita incorporar más instrumentos a la misma.

Una buena tarjeta de sonido, además de incluir la tecnología WaveTable, debe permitir que se añada la mayor cantidad posible de memoria. Algunos modelos admiten hasta 28 Megas de RAM (cuanta más, mejor).

Efectos.- Una tarjeta de sonido también es capaz de manipular las formas de onda definidas; para ello emplea un chip DSP (Digital Signal Processor, Procesador Digital de Señales), que le permite obtener efectos de eco, reverberación, coros, etc. Las más avanzadas incluyen funciones ASP (Advanced Signal Processor, Procesador de Señal Avanzado), que amplía considerablemente la complejidad de los efectos. Por lo que a mayor variedad de efectos, más posibilidades ofrecerá la tarjeta.

Polifonía.- ¿Qué queremos decir cuando una tarjeta tiene 20 voces? Nos estamos refiriendo a la polifonía, es decir, el número de instrumentos o sonidos que la tarjeta es capaz de emitir al mismo tiempo. Las más sencillas suelen disponer de 20 voces, normalmente proporcionadas por el sintetizador FM, pero hoy en día no debemos conformarnos con menos de 32 voces. Las tarjetas más avanzadas logran incluso 64 voces mediante sofisticados procesadores, convirtiéndolas en el llamado segmento de la gama alta.

MIDI.- La práctica totalidad de tarjetas de sonido del mercado incluyen puerto MIDI (Musical Instrumental Digital Interface, Interface digital de música instrumental); se trata de un estándar creado por varios fabricantes, que permite la conexión de cualquier instrumento, que cumpla con esta norma, al ordenador, e intercambiar sonido y datos entre ellos. Así, es posible controlar un instrumento desde el PC, enviándole las diferentes notas que debe tocar, y viceversa; para ello se utilizan los llamados secuenciadores MIDI.

En este apartado hay poco que comentar. Simplemente, si vamos a emplear algún instrumento de este tipo, habrá que cerciorarse de que la tarjeta incluya los conectores DIN apropiados para engancharla al instrumento en cuestión, y el software secuenciador adecuado, que también suele regalarse con el periférico.

Frecuencia de muestreo.- Otra de las funciones básicas de una tarjeta de sonido es la digitalización; para que el ordenador pueda tratar el sonido, debe convertirlo de su estado original (analógico) al formato que él entiende, binario (digital). En este proceso se realiza lo que se denomina muestreo, que es recoger la información y cuantificarla, es decir, medir la altura o amplitud de la onda. El proceso se realiza a una velocidad fija, llamada frecuencia de muestreo; cuanto mayor sea esta, más calidad tendrá el sonido, porque más continua será la adquisición del mismo.

Resumiendo, lo que aquí nos interesa saber es que la frecuencia de muestreo es la que marcará la calidad de la grabación; por tanto, es preciso saber que la frecuencia mínima recomendable es de 44.1 KHz, con la que podemos obtener una calidad comparable a la de un disco compacto.

Otras consideraciones.- Existen otros factores que se deben tener en cuenta: por ejemplo, mucha gente prefiere controlar el volumen de la tarjeta de forma manual, mediante la típica ruedecilla en la parte exterior de la misma. Sin embargo, la tendencia normal es incluir este control (además de otros, como graves, agudos, etc.) por software, así que se debe tener en cuenta este detalle si es importante para nosotros.

La popularización de Internet ha propiciado la aparición de un nuevo uso para las tarjetas de sonido: la telefonía a través de la red de redes. Efectivamente, con un micrófono y el software adecuado, podemos utilizar la tarjeta para hablar con cualquier persona del planeta (que posea el mismo equipamiento, claro) a precio de llamada local.

Sin embargo, la calidad de la conversación dependerá de dos conceptos: half-duplex y full-duplex. Resumiendo un poco, full-duplex permite enviar y recibir información al mismo tiempo, mientras que half-duplex sólo puede realizar una de las dos operaciones en cada momento. Traduciendo esto a una conversación, tenemos que el half-duplex nos obliga a hablar como si utilizáramos un walkie-talkie; es decir, hay que esperar a que uno diga algo para poder responder, mientras que el full-duplex nos ofrece bi-direccionalidad, es decir, mantener una conversación normal como si fuera un teléfono.

En algunos casos, el fabricante posee controladores que añaden funcionalidad full-duplex a tarjetas que no implementan esta forma de trabajo, por lo que puede ser una buena idea ir a la página Web del fabricante en cuestión.

Por último, y aunque sea de pasada, puesto que se trata de un requisito casi obligatorio, resaltaremos la conveniencia de adquirir una tarjeta que cumpla al cien por cien con la normativa Plug and Play; seguro que muchos lo agradecerán.

2.2.6. COMUNICACIÓN DEL COMPUTADOR CON DISPOSITIVOS EXTERNOS: PUERTOS SERIAL, PARALELO, USB.

Se puede conectar dispositivos externos a los conectores de entrada / salida (E/S). El BIOS (basic input/output system [sistema básico de entrada/salida]) del ordenador detecta la presencia de dispositivos externos cuando se inicializa o reinicializa su ordenador.

Se podría mencionar para nuestro tema:

Conector paralelo.- Se utiliza el conector paralelo de 25 agujeros para conectar un dispositivo paralelo al ordenador. El conector paralelo se utiliza principalmente para impresoras. También puede conectar la unidad de disquete al conector paralelo.

El puerto paralelo envía y recibe datos en formato paralelo, donde ocho bits (un byte) de datos se envían simultáneamente a través de ocho líneas separadas. El puerto puede configurarse como un puerto unidireccional (únicamente salida) para dispositivos como una impresora, o como un puerto bidireccional para dispositivos como un adaptador de red.

El puerto paralelo integrado del ordenador está designado como LPT1. Los sistemas operativos Microsoft® Windows® 95 y Windows 98 reconocen automáticamente el dispositivo paralelo y lo configuran correctamente. El puerto paralelo también puede configurarse para ser compatible con la norma PS/2.

Conector de acoplamiento.- El conector de acoplamiento da soporte a las soluciones de acoplamiento del APR (Apache Portable Runtime, Carrera temporal portátil Apache) de la familia C/Port y la Estación de expansión de la familia C/Dock de Dell.

Conector USB.- El conector USB soporta la utilización de un dispositivo concentrador USB para conectar dispositivos múltiples. Los dispositivos USB generalmente son periféricos de baja velocidad como ratones, teclados, impresoras y altavoces para ordenador. El APR de la familia C/Port y la estación de expansión de la familia C/Dock tienen dos conectores USB.

Conector serie.- Se utiliza el conector de 9 patas para instalar un dispositivo serie al ordenador.

El puerto serie pasa datos en formato serie (un bit a la vez a través de una línea). Este puerto soporta una gran variedad de dispositivos que requieren transmisión de datos seriales, incluido un ratón serie, impresora serie, plotter o módem externo.

Conector de alimentación de CA.- Se utiliza el conector de alimentación de CA para conectar el adaptador de CA al ordenador. El adaptador de CA convierte la alimentación de CA a la alimentación de CC requerida por el ordenador.

Puede conectar el adaptador de CA cuando su ordenador esté encendido o apagado.

El adaptador de CA funciona en enchufes eléctricos en todo el mundo. Sin embargo, los conectores de alimentación varían de un país a otro. Antes de utilizar una fuente de alimentación de CA en el extranjero, es posible que necesite obtener un nuevo cable de alimentación diseñado para utilizarse en ese país.

Puerto infrarrojo.- El puerto infrarrojo (IR) del ordenador es compatible con los estándares IrDA 1.1 (Fast IR [IR rápido]) y 1.0 (Slow IR [IR lento]). Un puerto infrarrojo le permite transferir archivos de su ordenador a otro dispositivo compatible con infrarrojos usando o sin usar cables. Una luz infrarroja se transmite a través de un lente en el ordenador a una distancia de hasta 1 m (3,3 pies). Esta luz es recibida por un ordenador, una impresora, un ratón o un control remoto compatible.

La dirección predeterminada del puerto infrarrojo es COM3. Para evitar conflictos por recursos con otros dispositivos, se debe reasignar la dirección del puerto infrarrojo.

Cuando el ordenador se está usando con el APR de la familia C/Port o la Estación de expansión de la familia C/Dock, el puerto infrarrojo en el ordenador queda automáticamente desactivado.

Se debe orientar el puerto infrarrojo del ordenador directamente hacia el puerto infrarrojo del dispositivo compatible. Los dispositivos infrarrojos transmiten datos en un cono de 30 grados de radiación infrarroja. Se debe iniciar el software de comunicaciones de datos en los dos dispositivos y después se empieza a transferir archivos. Es necesario asegurarse de leer la documentación incluida con el dispositivo compatible para utilizarlo correctamente.

De este proyecto surgieron dos redes: Una de investigación, ARPANET, y una de uso exclusivamente militar, MILNET. Para comunicar las redes, se desarrollaron varios protocolos. El protocolo de Internet y los protocolos de control de trasmisión. Posteriormente estos protocolos se englobaron en el conjunto de protocolos TCP /IP.

En 1980, se incluyó en el UNX 4.2 de BERKELEY, y fue el protocolo militar standard en 1983. Con el nacimiento en 1983 de INTERNET, este protocolo se popularizó bastante, y su destino va unido al de Internet. ARPANET dejó de funcionar oficialmente en 1990.

Algunos de los motivos de su popularidad son:

- Independencia del fabricante.
- Soporta múltiples tecnológicas.
- Puede funcionar en tecnologías de cualquier tamaño.
- Estándar de Estados Unidos desde 1983.

La arquitectura de un sistema en TCP /IP tiene una serie de metas:

- La independencia de la tecnología usada en la conexión a bajo nivel y la arquitectura del ordenador.
- Conectividad Universal a través de la red.
- Reconocimientos de extremo a extremo.
- Protocolos estandarizados.

2.3.1.2 LA PRÓXIMA GENERACIÓN IP

Inpg es también conocido como IP versión 6, es la nueva versión de IP (Internet Protocol), diseñada como sucesora de la actual versión de IP versión 4 (IPv4). Los cambios de la versión 6 con respecto a la versión 4 fueron incluidos para resolver las limitaciones de IPv4 e incluye:

- Aumentar la longitud de las direcciones a 128 bits, en lugar de 32 para soportar un rango de direcciones mucho más amplio y más flexibilidad en el direccionamiento.
- Facilidad de configuración que permite a los hosts autoconfigurar sus parámetros de red y su aplicación IP.
- Más niveles de jerarquía para reducir el tamaño de las tablas de enrutamiento y como consecuencia, más eficiencia con menos memoria.- Extensiones de enrutamiento con las cuales se pueden soportar nuevas funcionalidades como:

  - **Selección de proveedores.-** Una opción que permite a la máquina origen listar los nodos intermedios necesarios para alcanzar el destino.
  - **Máquinas móviles.-** Esta función permitirá conectar una máquina a la red sin la necesidad de configuraciones manuales. También son llamadas plug--and-play.
  - **Redirección automática.-** El destino puede responder a la dirección origen invirtiendo la secuencia de direcciones, eliminando así el proceso de enrutamiento.

2.3.2. PROCESAMIENTO DIGITAL DE VOZ

2.3.2.1. INTRODUCCIÓN

La integración de los sistemas de comunicación, junto con el constante crecimiento de la digitalización de las redes que utilizan dichos sistemas, hacen necesario una administración eficiente de los recursos disponibles. Uno de estos recursos, corresponde a la capacidad de memoria de almacenamiento de datos. Más específicamente, se busca que dicha capacidad sea maximizada, a través de la reducción de los paquetes de información almacenados. Es así como surge la compresión de datos como una de las posibles soluciones (*y quizás la más lógica*) al problema planteado en la administración eficiente de la capacidad de memoria.

La búsqueda de métodos de compresión de datos está ligada al tipo de datos almacenados, entendiéndose por *tipo* al origen de dichos datos. Dentro de estos tipos de datos, tenemos los datos procedentes del proceso de digitalización de la voz humana. A su vez, dichos datos pueden ser nuevamente clasificados según el método de digitalización de voz efectuado.

Dentro de las aplicaciones de la compresión de audio, se distinguen 4 áreas: Broadcasting (difusión), almacenamiento, multimedia y telecomunicaciones. Ejemplos de estos son: almacenamiento en disco (CD audio, video, etc.), televisión por cable y satelital, Internet (audio y video "streaming"), aplicaciones ISDN (Integrated Services Digital Network, Redes Digitales de Servicio Integrado), etc.

reducción y cancelación de ruido, la utilización de modelos auditivos para la representación de la señal de voz en sistemas de reconocimiento del tabla, así como el desarrollo de algoritmos de detección de voz / silencio, pitch y sonoridad.

2.3.2.6 SÍNTESIS DE VOZ

El proceso de síntesis de voz dota a las máquinas de la capacidad de producir mensajes orales no gravados previamente como es el caso de los sistemas de respuesta oral.

Tomando como entrada cualquier texto, los sistemas de síntesis de voz realizan al proceso de lectura de forma clara e inteligible y como una voz lo más natural humana posible. La síntesis de voz conforma el interfaz oral de comunicación entre una máquina y el usuario de la misma. Las investigaciones se encaminan en el desarrollo de un sistema de conversión texto – voz por concatenación de unidades.

2.3.2.7 CARACTERÍATICAS DE LOS PROCESOS DIGITALES

El procesamiento digital de señales analógicas se basa en la suposición de que las señales bajo consideración pueden representarse en función de los valores que toman en un conjunto discreto de puntos.

El teorema de muestreo de C. E. Shannon en 1949 marca las restricciones para las frecuencias de una señal que es función del tiempo, f(t), y puede ser resumido como sigue:

A fin de que una señal f(t) pueda ser recuperada de la forma más fiel posible, es necesario que la frecuencia de muestreo sea de por lo menos el doble de la componente más alta de frecuencia.

Las consecuencias del muestreo de una señal con frecuencia menor a su componente más alta es un fenómeno conocido "aliasing". Este concepto resulta de una toma de muestras erróneas y todo termina en la obtención de una señal totalmente distinta, cuando se trata de recuperar la señal original.

2.3.3. VOZ SOBRE IP

2.3.3.1. INTRODUCCIÓN

Voz sobre IP, o VoIP, es la última tecnología hecha para la computación y telefonía en los últimos 30 años. Se trata de la tecnología que permite comprimir el sonido, empaquetarlo y transmitirlo sobre la red IP, para ser recibido por equipo en el otro lado que convierte los paquetes de nuevo en sonido. Lo que esto significa efectuar llamadas o enviar faxes entre las diferentes localidades de una oficina sobre la red IP de la Intranet, la cual utiliza para la transmisión de datos.

2.3.3.2. PRESENTACIÓN DE LA TECNOLOGÍA

La tecnologia inicial, empezó con las tarjetas de expansión I/O (input / output) de las PCs, en una exposición reciente llevada a cabo en los Ángeles, no menos de 100 expositores mostraban productos de empaquetamiento sobre la plataforma NT 4.0 con estas tarjetas de "Voz sobre IP" y bastante software de instalar una red de voz con la poca confiabilidad que ofrecen los productos telefónicos con 10 o 20 años de probabilidad de fallo y no hace falta especificar que las computadoras o cualquier computadora, sobre cualquier sistema operativo pueda asegurar esta clase de seguridad.

Equipos con este tipo de sistema, están hoy en día llegando a los mercados con esta tecnología; siendo los mayores productores: Cisco, Lucent, Siemens, Micom y otros. El equipo consta de dos partes. La primera que es una "Tarjeta de Interface de línea" que conecta su teléfono, fax, sistema de teclado, o conmutador (PBX). Se comporta como un circuito de línea telefónica.

La segunda parte del equipo entra en acción, y digitalizada el sonido y lo comprime en pequeños paquetes, envolviéndolos de tal manera que el paquete se vea reconocido como voz mientras se transmite en la red de transmisión de datos.

Ahora, después de un año, Cisco se encuentra con la posibilidad de empezar a ofrecer parte de los mejores productos de integración de voz y datos. La forma de proceder de Cisco es muy simple: Empaquetada la interface de voz y los productos que se encargan de la comprensión de datos dentro del ruteador. Los Ruteadores Cisco tienen una esperanza de vida de entre 10 y 25 años. Y este es el tipo de confiabilidad que esperamos de nuestras redes de voz, obviamente el costo para la adquisición de estos equipos es elevado.

2.3.3.3 VOIP ES VOZ SOBRE EL INTERNET

Sin duda alguna vez usted leerá sobre "voz a través del Internet". Pero esto es un error, los productores elaboran equipos que pueden enviar voz sobre redes basadas en el protocolo IP. Obviamente que el Internet está basado en el IP, otro factor a considerar es el sobre - estimación que ha tenido el Internet y la pobre confiabilidad de la red es muy variable, en ocasiones es rápido y en otras es lento. La voz requiere de una red con velocidad asegurada y no variable. La alta calidad que requiere el servicio de voz asegura todos los segmentos en los que viaja la información y lleguen al oído de la persona que escucha, a la misma velocidad, segmento por segmento. Si existen retrasos o paquetes perdidos, se pierden las sílabas la calidad de sonido se vuelve irregular.

Es aquí donde especificaremos un nuevo término "red IP controlada". Una red IP controlada que el usuario pueda monitorear y regular para asegurar que todos los paquetes de transmisión lleguen a una velocidad de por lo menos 180 ms (milisegundos) desde el equipo en que se origina la llamada al equipo que la recibe.

2.3.3.4. TRANSMISIÓN DE VOZ SOBRE IP

IP ha tenido su origen en transmisión de datos y no está adaptado a la transmisión de datos e imágenes. La tecnología de transmisión de paquetes, en la que está basada IP, ofrece tamaño de celdas variable, que en comparación con tecnologías de tamaño de celda fija como ATM (Asynchronous Transfer Mode, Modo de Tranferencia Asincrónica), introduce ineficiencias y necesidad de proceso extra. Además IP es un protocolo que solamente ofrece un tipo de calidad y servicio (QoS) basado en proporcionar el mejor rendimiento posible en el enlace disponible.

El estándar H.323 se define como el standard que permite el tráfico multimedia, en tiempo real sea intercambiado sobre una red de paquetes, tal y como es una red IP, añadiendo también la capacidad de flujos multimedia (retransmisiones de audio o video). H.323 define una serie de entidades en una red H.323 con una serie de funcionalidades tales como:

Proxies: Son los sistemas que actúan con intermediarios entre diversas entidades, tal y como lo hacen los proxies en las redes IP (conexión entre la Intranet e Internet, por ejemplo).

El establecimiento y el mandamiento de conexiones H.323 realizan un uso tanto de tráfico sobre TCP como UDP:

- Q.931 sobre TCP que se realiza a través del puerto bien conocido 1720 para negociar el puerto de conexión de H.245.
- H.245 sobre TCP para realizar las negociaciones de los parámetros (codificadores entre otros) y realiza las conexiones UDP para RTP y RTCP.
- RTP y TTCP sobre UDP en que usan conexiones UDP para mantener los flujos asociados con el tráfico H.323.

El standard H.323 define un método de permitir tráfico multimedia sobre una red IP, pero y como no puede de otra forma, no asegura que la comunicación pueda tener lugar. En el caso de transmisión de voz es necesario asegurar unos parámetros mínimos para que una conversación pueda tener lugar. Los parámetros más influyentes en el comportamiento de una transmisión de voz son los siguientes:

Retardos de los paquetes: Una red IP, y sobre todo Internet, no asegura el retardo de un paquete. Actualmente, solamente a través del control y gestión global extremo a extremo, y la disponibilidad de suficiente ancho de banda así como la tecnología de switching - routing necesaria, es posible asegurar unos niveles de retardo máximos. Por ello y en el estado de congestión actual y previsible, Internet no nos puede asegurar unos niveles máximos.

Jitter: Es muy importante del retardo de los paquetes, y consiste en el tiempo de variación en la llegada de paquetes. Este parámetro tiene los mismos problemas y dificultades que el retardo, por lo que las soluciones van en la misma línea. Si cabe, en este caso son más importantes las tecnologías de enrutamiento de los paquetes IP.

Pérdida de Paquetes: Al estar basados, sobre todo UDP, en una transmisión no fiable las pérdidas de paquetes si existe congestión o problemas en la transmisión pueden llegar a ser importantes.

El estado de la red tiene un impacto diferente sobre la transmisión de fax (protocolos T.4 y T.30) sobre IP que sobre la transmisión de voz. El oído humano es mucho más sensible a la pérdida de datos, que puede hacer la conversación ininteligible, que al retardo. LA UIT ha desarrollado una recomendación para ayudar a definir los defectos de los retardos dando un valor máximo. La recomendación G.114 definida en 1996 recomienda que el límite en un canal unidireccional de voz sea de 400 ms. Sin embargo tenemos que considerar que la apreciación de la cantidad de una comunicación de voz tiene una buena parte subjetiva, dependiendo también de valor calidad / precio que se le de a esa comunicación. Puede que retardos de 400 ms resulten inadmisibles para una buena parte de los usuarios para conversaciones de negocios, y que retardos de 600 ms resulten admisibles por usuarios privados si el costo así se lo justifica.

La pérdida de paquetes también afecta a la calidad de voz, pero el tanto por ciento admisible depende tanto de los algoritmos de comprensión usados, algunos son capaces de recuperar errores, como de la percepción subjetiva de los usuarios. El límite generalmente aceptado como máximo se sitúa alrededor del 8 10%.

La realidad es que el asegurar estos parámetros, esta calidad de servicio, a lo largo de una red IP con los niveles de calidad habituales en una red de voz, sólo es posible, y con limitaciones, cuando se realiza dentro de una red IP privada con los equipos y el ancho de banda necesarios y siendo gestionada centralizadamente. Habitualmente un canal de voz necesita un ancho de banda garantizado de 12 15 Kb/s por lo que proporcionar o asegurar en una red como Internet ese ancho de banda no es posible en general. La utilización de las nuevas redes IP por los operadores puede hacer posible la disponibilidad, dentro de esas redes IP, de ancho de banda garantizado; pero sin duda, con el costo asociado de reserva de ese ancho de banda. La comparación de las conexiones tanto para datos como voz IP reducirá los costos globales, pero no se puede suponer que si se desea obtener una calidad comparable a la que la red de voz tiene, los costos se reduzcan muy significativamente. La tendencia a la reducción del precio del ancho de banda, así como la integración de servicios reducirán los costos de conexiones, pero el aseguramiento de calidades de servicio tendrá su costo, aunque menor.

2.3.3.5. EL ESTÁNDAR VOZ SOBRE IP (VOIP)

Desde hace tiempo, los responsables de comunicaciones de las empresas tienen en mente la posibilidad de utilizar su infraestructura de datos, para el transporte del tráfico de voz interno de la empresa. No obstante, es la aparición de nuevos estándares, así como la mejora y abaratamiento de las tecnologías de comprensión de voz, lo que está provocando finalmente su implantación.

Después de haber constatado que desde un PC con elementos multimedia, es posible realizar llamadas telefónicas a través de Internet, podemos pensar que la telefonía en IP es poco mas que un juguete, pues la calidad de voz que obtendremos a través de Internet es muy pobre. No obstante, si en nuestra empresa disponemos de una red de datos que tenga un ancho de banda bastante grande, también podemos pensar en la utilización de esta red para el tráfico de voz entre los distintos departamentos de la empresa o una entidad cualquiera. Las ventajas que se obtendrán al utilizarla en una red para transmitir tanto voz como los datos son evidentes.

VENTAJAS.-

Las principales ventajas de la voz sobre IP son:

- La de instalación y cableado.
- La de movilidad de los puestos
- La posibilidad de remotizar puestos.
- Ahorro de costos de comunicaciones pues las llamadas entre los distintos departamentos o unidades dentro de una empresa o entidad saldrían gratis.
- Integración de servicios y unificación de estructura.

DESVENTAJAS.-

Los inconvenientes de la voz sobre IP son:

- Que puede haber empeoramiento en la calidad de la voz.
- Que hay que controlar el trafico en la red local (LAN).
- Al ocupar un ancho de banda constante el número de operadores puede estar limitado.

Realmente la integración de lo voz y los datos en un misma red es una idea antigua, pues desde hace tiempo han surgido soluciones desde distintos fabricantes que, mediante el uso de multiplexores, permiten utilizar las redes WAN de datos de las empresas (típicamente conexiones punto a punto y frame - relay) para la transmisión del tráfico de voz. La falta de estándares, así como el largo plazo de amortización de este tipo de soluciones no ha permitido una amplia implantación de las mismas.

Es innegable la implantación definida del protocolo IP los ámbitos empresariales a los domésticos y la aparición de un estándar, el VoIP, no podía hacerse esperar. La aparición del VoIP junto con el abaratamiento de los DSP's (Procesador Digital de Señal), los cuales son claves en la comprensión de la voz, son los elementos que han hecho posible el despegue de estas tecnologías. Para este auge existen otros factores, tales como la aparición de nuevas aplicaciones o la apuesta definitiva por VoIP de fabricantes como Cisco Systems o Nortel – Bay Networks. Por otro lado los operadores de telefonía están ofreciendo o piensan ofrecer en un futuro cercano, servicios IP de calidad a las empresas.

Dentro de la tecnología, podemos encontrar 3 tipos de redes:

1. ***Internet.***- El estado actual de la red no permite un uso profesional para el tráfico de voz.
2. ***Red IP pública.***- Los operadores ofrecen a las empresas la conectividad necesaria para interconectar sus redes de área local en lo que al tráfico IP se refiere. Se puede considerar como algo similar a Internet, pero con una mayor calidad de servicio y con importantes mejorar en seguridad. Hay operadores que incluso ofrecen garantías de bajo retardo y / o ancho de banda, lo que las hace muy interesante para el tráfico de voz.
3. ***Intranet.***- La red IP implementada por la propia empresa. Suele constar de varias redes LAN (Ethernet conmutada, ATM, etc.) que se interconectan mediante redes WAN tipo Frame Relay / ATM, líneas punto a punto, RDSI para el acceso remoto, etc. En este caso la empresa tiene bajo su control prácticamente todos los parámetros de la red, por lo que resulta ideal para su uso en el transporte de la voz.

El VoIP tiene como principal objetivo asegurar la interoperabilidad entre equipos de diferentes fabricantes, fijando aspectos tales como la supresión de silencios, codificación de la voz y direccionamiento, y estableciendo nuevos elementos para permitir la conectividad con la infraestructura telefónica tradicional.

Podemos resumir diciendo que VoIP es una tecnología que tiene todos los elementos para su rápido desarrollo. Como muestra ver compañías como Cisco, la han incorporado a su catálogo de productos, los teléfonos IP están ya disponibles y los principales operadores mundiales, así como Telefonía, están promoviendo activamente el servicio IP a las empresas, ofreciendo calidad de voz a través del mismo. Por otro lado tenemos ya un estándar que nos garantiza interoperabilidad entre los distintos fabricantes. La conclusión parece lógica: hay que estudiar cómo podemos implantar VoIP en nuestra empresa

2.3.3.6. VOZ SOBRE IP EN RED LOCAL

En un Drama Call Center se puede eliminar toda la red de telefonía y hacer que las conversaciones circules por la red (LAN) de datos. En los PCs de los operadores, se puede utilizar un programa de CTI para hacer "Phone - Scren" y utilizar una simulación de puesto telefónico en la pantalla del PC. Los auriculares del operador se conectan a la tarjeta de sonido del PC.

En cualquier punto donde llegue la red de datos de una empresa, se puede conectar un puesto de operadora del Centro de Llamadas de la Empresa.

Al ir la voz sobre protocolo IP puede circular por la red informática, routers, hubs, modems, etc.

2.3.3.7. VOZ SOBRE IP ENTRE CENTROS

Cuando existe una conexión de datos entre centros de una empresa, se puede hacer circular por ella las conversaciones de voz, y los paquetes de datos que implican los comandos CTI de la simulación de un terminal telefónico.

Normalmente los anchos de banda son mucho más reducidos que en una red local de datos, por lo que la voz se ha de comprimir, y reducirla de 64 Kbits a 5,3 Kbits para que ocupe poco ancho de banda en la transmisión. Esta comprensión se hace con los Gateways de DataVoice. Se puede unir varios Pc a un centro de Llamadas o unir dos centros entre sí.

No hace muco tiempo, ATM era visto por todos los operadores de telecomunicaciones como la única tecnología integradora de todo tipo de tráficos: datos, video y por supuesto del tráfico de voz. Sin embargo, ATM ha visto como su desarrollo e implantación han ido más lentos de lo esperado y su extensión sobre todo el entorno LAN está en duda. A la vez, IP surgido como un protocolo de LAN de transmisión de datos, ha ido extendiéndose hacia la MAN y la WAN de un modo imparable debido en parte a su sencillez, debido en parte a su bajo costo tanto en equipos como en transporte tanto a través de redes IP como de Internet.

2.3.3.8. REDES DE VOZ CORPORATIVAS SOBRE IP

Las actuales grandes redes de voz corporativas están basadas en PBX, bien sean Ericsson, Alcatel, Siemens, Nortel, Lucent, Philips, MATRA, con sistemas con miles de extensiones con conexiones y servicios comunes. Todos estos sistemas a través de protocolos propietarios o comunes como Q.SIG se unen con enlaces de 2 Mb /s punto a punto, ya sean medios privados o medios públicos. Con objeto de poder proporcionar distribuidamente las facilidades de las PABX estas uniones dialogan en protocolos como DPNSS, CORNET o un standard Q:SIG que necesitan unos parámetros de retardos y anchos de banda para mantener la señalización y los enlaces.

Tradicionalmente el costo de los medios de transmisión punto a punto ha sido alto, por lo que el costo de la unión de las PABX remotas ha sido siempre muy alto, por lo que cuando el tráfico no es muy intenso, no se ha justificado la unión de 2 Mb / s, y las llamadas se han realizado sobre la red Pública con sus costos asociados.

La comparación de medios para la reducción de costos siempre ha sido un objetivo en las grandes redes. La Red de Datos y la Red de Voz tradicionalmente han estado completamente separados por lo que la evolución tanto de los protocolos de unión como de los medios de unión han sido diferentes, así tanto las tecnologías para interconexión como gestión están separadas en la actualidad.

ATM puede ser la primera tecnología ampliamente implementada que fue diseñada pensando en la integración de servicios de voz, datos y video sobre la misma red. ATM es un protocolo completo

mediante redes publicas IP (dificilmente sobre la Internet actual) no sólo con el objetivo de reducción de costos sino como nueva aplicación de la unión e interconexión de dispositivos multimedia H.323. La evolución hacia IP no se dirige únicamente a la sustitución de las actuales redes de voz y datos, sino que durante un largo tiempo (la definición de "largo" en el mundo de las comunicaciones parece difícil) convivirán varias redes que se interconectarán y se comunicarán entre ellas, haciendo una migración paulatina del volumen de tráfico desde la voz standard o los datos tipo host hacia el tráfico multimedia basado principalmente sobre IP. En estos momentos la interconexión de voz puede cubrir dos necesidades:

- Reducción de costos de las comunicaciones de larga distancia de voz a través de la utilización de la infraestructura de datos, pero a cambio de una posible falta de aseguramiento de calidad.
- Nuevas aplicaciones de interconexión de dispositivos multimedia con las redes actuales de voz.

El escenario que podemos obtener se basa ya en una multitud de productos de voz sobre IP que permite una mezcla de las redes de voz y datos actuales y la telefonía sobre IP. Todos estos productos utilizan siempre como tanto los enlaces TTB como IP para la interconexión, enrutando la llamadas cuando la conexión sobre IP no es posible o no da la calidad requerida en ese momento. Son de destacar los equipos de Lucent y de Nortel que proporcional unas posibilidades reales de implementación (Ericsson, Siemens, Microsoft, PictureTel, 8 X 8, Radvision, Videoserver, Vocaltec, British Telecom., Teles, First Virtual,... soportan H.323 también).

Como conclusión podemos asegurar que la voz sobre IP ya es posible, que la evolución de su uso vendrá con la evolución tanto de la infraestructura de transporte como del protocolo y que en la actualidad las diversas implementaciones tienen como objeto tanto el ahorro como de costos como el proporcionar nuevos servicios tanto en lugar como en funcionalidad.

2.3.3.9. INTEGRACIÓN DE VOZ, DATOS Y VÍDEO

En la empresa clásica, la red de voz (telefonía es diferente de la red de datos (LAN / Router) y de existir, diferente de la red de videoconferencia que en muchos casos va sobre RDSI. Las empresas que utilicen esta estructura, pueden obtener un considerable ahorro en sus gastos de comunicaciones, integrando (sumando) los diferentes tipos de tráfico sobre un sistema de gestión del ancho de banda (passport). La salida de este se puede conectar, por ejemplo a una red Frame Relay o ATM pública, o a una red IP con calidad de Servicio, etc. Las cuotas fijas que anteriormente se pagaban por cada de las diferentes redes, ahora se convierten en una sola, con una gestión única.

2.3.3.10. CONSOLIDACIÓN DE SERVICIOS MULTIMEDIA SOBRE REDES IP

Los elementos que tradicionalmente han formado parte de las comunicaciones empresariales (centrales telefónicas, redes de datos, sistemas de videoconferencia,...) evolucionan hacia un nuevo escenario de integración total o consolidación.

La evolución de las tecnologías de la información nos permite introducirnos en una nueva era audiovisual en la que el término "comunicación" quiere decir algo más que intercambiar documentos o datos, hablar, ver, videoconferencias. Una época, en la que tomar decisiones es mucho más sencillo pues se usan los mecanismos "humanos" de comunicación junto con la potencia y flexibilidad que aportan las nuevas tecnologías.

Factores clave de esta consolidación son: la implantación definida del protocolo IP desde los ámbitos empresariales a los domésticos, la convergencia de la tecnología ATM e IP y la aparición de estándares como VoIP (Voz sobre IP), H.323 (Videoconferencia en LAN), etc. Otro factor, es la apuesta definitiva por VoIP de fabricantes como Cisco Systems o Nortel - Bay Networks. Por otro lado los operadores de telefonía e ISPs están ofreciendo o piensan están ofreciendo o piensan ofrecer en un futuro cercano, servicios IP de calidad a las empresas.

2.3.3.11. GESTIÓN Y SEGURIDAD DE LA RED Y LAS COMUNICACIONES

Las redes de datos y de comunicaciones son la base de los sistemas de información y uno de los elementos más críticos del engranaje corporativo. La gestión de este recurso se convierte en un factor muy importante, ya que contribuye a su mantenimiento, a la resolución y prevención de incidencias, y en definitiva a asegurar la máxima disponibilidad.

La gestión activa puede mejorar el rendimiento de la red detectando "cuellos de botella", debido a saturación de tráfico en ciertos segmentos, o a un dimensionado inadecuado de algún componente de la red, como puede ser el acceso a la misma de los servidores.

La gestión permite la definición de alarmas que actúan en respuesta a eventos específicos (caídas de nodos, superación de umbrales error o tráfico). Estas alarmas pueden asociarse a acciones que pueden automatizar parte de la gestión. Esto facilita en gran medida el trabajo del administrador de la red al permitir la rápida detección y resolución de las incidencias.

Otro aspecto relacionado con la gestión es la generación automática de gráficos e informes que reflejan la situación de nuestra red en el día a día. Estos gráficos permiten ver la evolución del tráfico y sus tendencias, lo que facilita la anticipación a problemas futuros (congestión de nodos, problemas de saturación de interfaces de acceso, etc...). Esta radiografía de la red permitirá también ver el comportamiento de los diferentes servicios, es decir, cuales son los que están consumiendo más recursos, o cuales están creciendo más debido a la mayor demanda de los usuarios.

La accesibilidad a los servicios corporativos, tanto desde el interior como desde el exterior, afecta a la seguridad de la red corporativa. El sistema de seguridad es complejo, y se basa en el establecimiento de reglas (relaciones lógicas entre usuarios, redes y servicios) que validarán los datos de entrada y salida, bloqueando el tráfico no autorizado.

Las políticas de seguridad definen zonas desmilitarizadas y zonas protegidas, por lo que influyen directamente en la topología lógica de la red. Es por ello que, en la medida de lo posible, las tareas de gestión y seguridad tendrían que estar plenamente coordinadas.

Las plataformas de gestión se basan en aplicaciones tales como Network Node Manager de HP, a las que añaden los diferentes módulos de gestión que proporcionan los diferentes fabricantes, como pueden ser ForeView del fabricante FORE Systems, CiscoWorks, Optivity o Trancend. Los sistemas de seguridad se basan en aplicaciones como FireWall – 1, que es uno de los estándares de ipso en gestión de los dispositivos de seguridad para el acceso a redes.

El Gateway IPVox es la solución ideal para implementar servicios de Voz sobre IP (VoIP) sobre las redes de datos ya existentes en las empresas, produciendo un significativo ahorro en los costos en llamadas de larga distancia desde el primer momento, así como una optimización en el uso de las infraestructuras disponibles.

2.3.3.12. CALIDAD DE LA VOZ

La arquitectura de Switching MVIP integrada, la cual conecta las tarjetas de interfaz de línea (analógica o digital) con las tarjetas de compresión basadas en DSP, segura un gran rendimiento y una calidad de voz estable, punto que es crucial cuando se usan Gateways de VoIP.

También se producen mejoras adicionales sobre la calidad de voz con la cancelación de eco integrada y la supresión de silencio, característica esta última que optimiza el uso del ancho de banda necesario, ya que mientras se producen silencios debidos a las pausas en la conversación, no se transmite ningún paquete, liberando así dicho ancho de banda. Por otro lado, el control de ganancia puede ser variado voluntariamente, en lo que permite al administrador del sistema disponer de un control de volumen, optimizando así la calidad de la voz.

Varios tipos de CODEC pueden ser usados en la comprensión y descompresión de la voz para la transmisión sobre una red IP, entre los que se incluyen G.723, GSM y G.711. De esta manera se consigue que tan sólo se consuma un ancho de banda de aproximadamente 10 Kbps por llamada (incluyendo overhead), esto significa que con conexiones de 56 Kbps dispondríamos de recursos más que suficientes para mantener varias conversaciones simultáneamente.

CAPITULO III

PROYECTO DE ESTUDIO

3.1 HARDWARE DE INTERCONEXIÓN ENTRE PC Y PARLANTES DEL SISTEMA DE MULTIDIFUSIÓN DE MENSAJES DE VOZ

3.1.1 DIAGRAMA ESQUEMÁTICO

VER ANEXO 1

3.1.2 SALIDA DE DATOS DEL PC AL HARDWARE

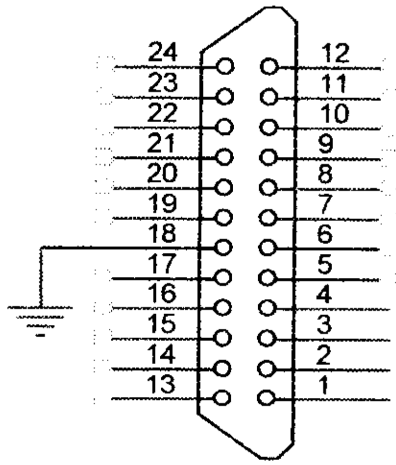


FIGURA 10: Pines existentes en el puerto paralelo de 1 a 24

Los datos que entrega el PC son datos binarios (1, 2, 4, 8, 16, 32,...128), para lo cual necesitamos un decodificador para que estos datos a su salida del mismo nos entreguen datos en decimal.

A la salida del puerto paralelo del computador nos entrega los siguientes datos:

Salida DB25				Salida Decodificador
pin2	pin3	pin4	pin5	
1	0	0	0	1
1	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	0	8
1	0	0	1	9

TABLA 1: Tabla de la salida de datos del puerto paralelo al computador

Si no estuviera el decodificador como observamos en la salida del DB25, si por ejemplo ingresamos el dato 3 mandaría un 0011 y en este caso encendería 2 parlante a la vez y el que queremos solo es el parlante 3,la gran ventaja con el decodificador es que podemos elegir la salida de audio que queramos del 1 al 9 y si quisiéramos en tiempos posteriores aumentar más salidas solo haría falta conectar otro decodificador a los pines del DB25(5,6,7,8) la desventaja de trabajar en decimal para este caso de audio es que solo podemos escoger una caja de audio a la vez, ya que internamente el integrado decodifica :suma los números binarios y nos entrega en decimal, este integrado trabajo con 5 Vcd, tiene 18 pines, el pin 8 que va a tierra y el pin 16 que es 5Vcc, los pines de entrada de los datos del DB25 van a los pines(15,14,13,12) y salen ya codificados por los pines(2,3,4,5,6,7,9,10,11) . Out dates (1 2 3 4 5 6 7 8 9)

Una vez ya con los datos en decimal necesitamos una interfaze para que controle cada caja de audio, interfaze significa trabajar con dos señales distintas sin que estas se mezclen en este caso con la salida mínima de voltaje del decoder hacer funcionar cada señal de audio de las cajas de los speakers.

3.1.3 RELE

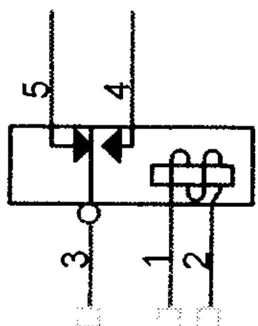


FIGURA 11: Estructura de un Relé

Relé es un dispositivo que nos sirve como interfaze para controlar señales distintas, en el pin 1 y 2 tenemos una bobina magnética y ésta al energizarse hace que se cierren los contactos 4(pin 4 y 5) haciendo posible así la circulación de la señal del pin 3 al pin 4 de este relé (3 y 4, se unen por la atracción magnética).

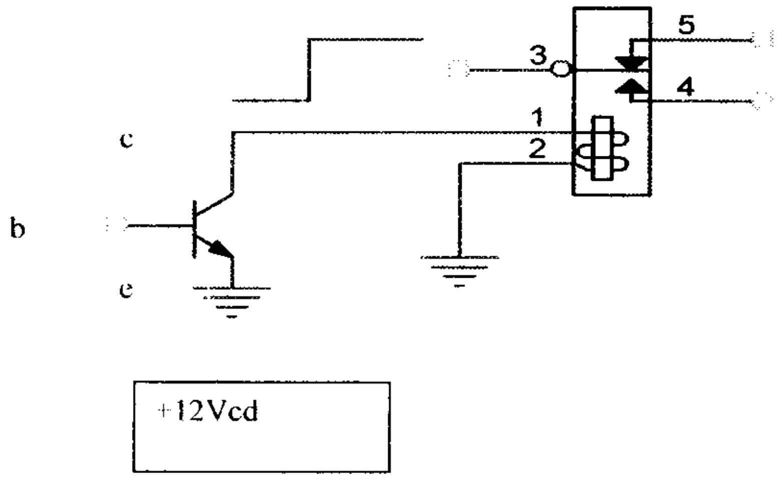


FIGURA 12: Transmisión de señal por el Relé

El pulso de voltaje proveniente del decodificador enviara la base del transistor, y este a su vez se satura o entra en conducción haciendo posible la circulación de corriente desde emisor al colector y energizándose de ésta manera la bobina y a su vez cerrando los contactos del pin 3 y 4, por el pin tres introducimos la señal de audio, y al momento de cerrarse con el contacto del pin 4 este deja pasar la señal de audio a la caja de parlante e inversamente Así de esta forma podemos controlar a cada caja de audio con ayuda de la interfaze de relés, estos trabajan con 12Vcd.

3.1.4 FUENTE DE 5VCD Y 12 VCD

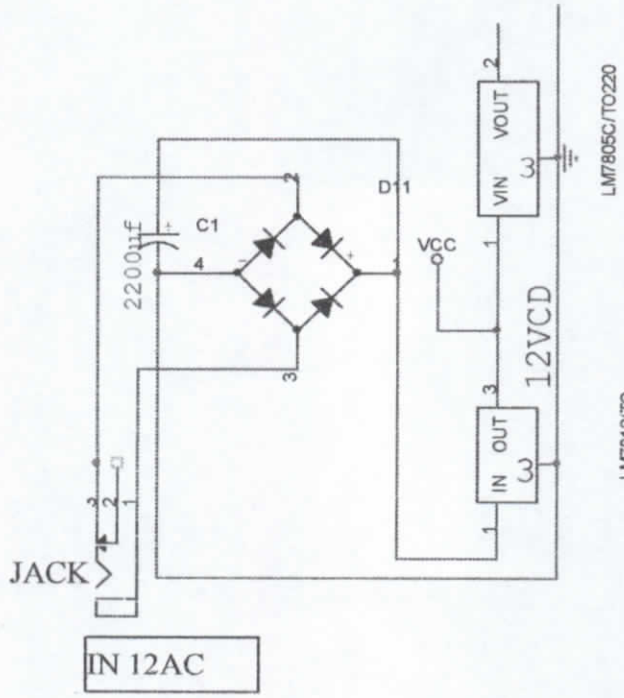


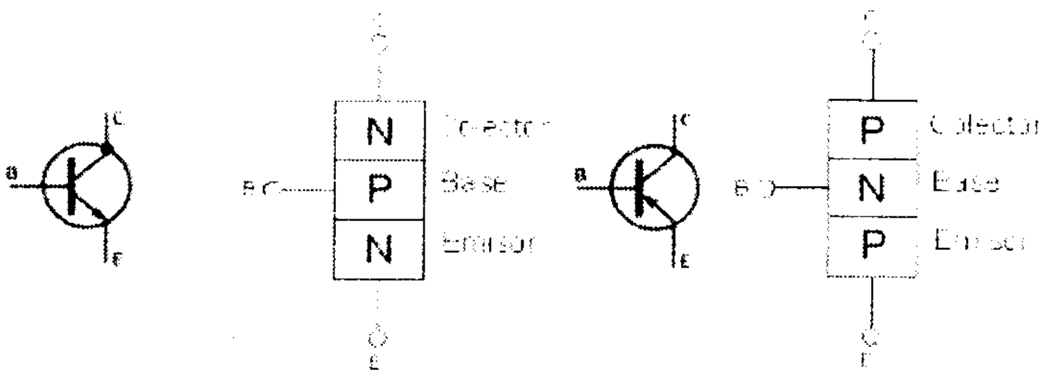
FIGURA 13: Regulador de voltaje

La corriente de 12AC entra por el jack éste es rectificada por el puente rectificador la que transforma en corriente directa VCD. El capacitor 2200uf elimina los picos que hayan quedado del rectificado, el integrado LM7812 solo deja pasar 12 vcd así entre entrado 17 o más vcd, este voltaje nos sirve para dar energía a la bobina del relé, para que se magnetice, así también el 7805 recibe los 12vcd y solo dejará pasar 5vcd. En los reguladores de voltaje, estos 5vcd sirven para dar corriente a la etapa del codificador y a los diodos leds (emisores de luz).

3.1.5 EL TRANSISTOR

Dispositivo semiconductor que permite el control y la regulación de una corriente grande mediante una señal muy pequeña. Existe una gran variedad de transistores. En principio, se explicarán los bipolares.

Los símbolos que corresponden a este tipo de transistor se muestran en la FIGURA 14:



Transistor NPN | Estructura de un transistor NPN | Transistor PNP | Estructura de un transistor PNP

FIGURA 14: Tipo de Transistores

Cuando el interruptor SW1 está abierto no circula intensidad por la Base del transistor por lo que la lámpara no se encenderá, ya que, toda la tensión se encuentra entre Colector y Emisor.

Generalmente podemos decir que la unión base - emisor se polariza directamente y la unión base - colector inversamente.

3.1.5.2 ZONAS DE TRABAJO

CORTE.- No circula intensidad por la Base, por lo que, la intensidad de Colector y Emisor también es nula. La tensión entre Colector y Emisor es la de la batería. El transistor, entre Colector y Emisor se comporta como un interruptor abierto.

SATURACION.- Cuando por la Base circula una intensidad, se aprecia un incremento de la corriente de colector considerable. En este caso el transistor entre Colector y Emisor se comporta como un interruptor cerrado. De esta forma, se puede decir que la tensión de la batería se encuentra en la carga conectada en el Colector.

ACTIVA.- Actúa como **amplificador**. Puede dejar pasar más o menos corriente.

Cuando trabaja en la zona de corte y la de saturación se dice que trabaja en ***conmutación***. En definitiva, como si fuera un interruptor.

$$\boldsymbol{\beta = I_C / I_B}$$

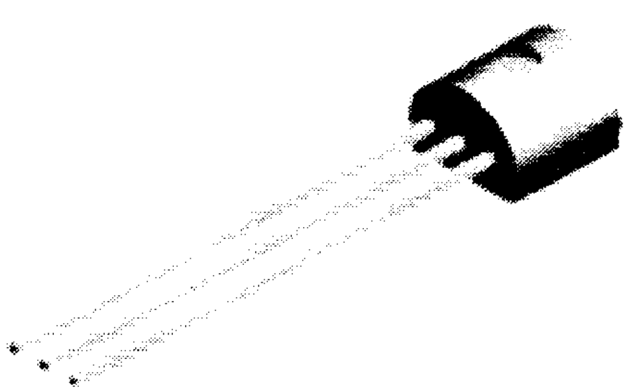
En resumen, se puede observar en la tabla 2

	Saturación	**Corte**	**Activa**
V_{CE}	≈ 0	$\approx V_{CC}$	Variable
V_{RC}	$\approx V_{CC}$	≈ 0	Variable
I_C	Máxima	$I_{CEO} \approx 0$	Variable
I_B	Variable	$= 0$	Variable
V_{BE}	$\approx 0{,}8$v	$< 0{,}7$v	$\approx 0{,}7$v

TABLA 2: Zona de trabajo del transistor

Los encapsulados en los transistores dependen de la función que realicen y la potencia que disipen, así nos encontramos con que los transistores de pequeña señal tienen un encapsulado de plástico, normalmente son los más pequeños (TO- 18, TO-39, TO-92, TO-226 ...); los de mediana potencia, son algo mayores y tienen en la parte trasera una chapa metálica que sirve para evacuar el calor disipado convenientemente refrigerado mediante radiador (TO-220, TO-218, TO-247...) ; los de gran potencia, son los que poseen una mayor dimensión siendo el encapsulado enteramente metálico . Esto, favorece, en gran medida, la evacuación del calor a través del mismo y un radiador (TO-3, TO-66, TO-123, TO-213...).

FIGURA 18: Grafico de un Transitor



3.1.6 INTEGRADO DECODIFICADOR

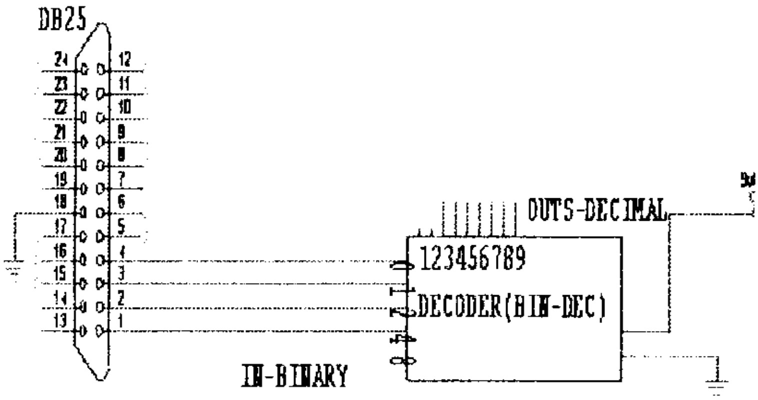


FIGURA 19: Decodificador (transforma de binario a decimal)

El integrado decodificador, codifica sus entradas provenientes del CPU, que vienen en binario a decimal y es gracias a ello que a la salida de este integrado tenemos datos independientes (de cero a nueve); Ejemplo: Si en el Software introducimos 4 en decimal a la salida del CPU por el puerto paralelo tenemos 100, en este caso el circuito integrado codifica éste dato y nos entrega el 4 en decimal por el pin o pata 4, y los otros pines o patas están en 0, y éste pulso es el que a su vez activa cada una de las etapas de audio. Si introducimos el 9 en decimal a la salida de CPU por el puerto paralelo tenemos 1001, en este caso el circuito integrado codifica este dato y nos entrega el 9 en decimal por el pin o pata 9, y los otros pines o patas están en 0, y este pulso a su vez activa cada una de las etapas de audio.

Y si queremos aumentar más etapas de audio simplemente tendremos que aumentar otro decodificador a la salida del puerto paralelo ya que para cada puerto del sistema hemos utilizado 1-2-4-8 de izquierda a derecha.

3.2 SOFTWARE DE INTERCONEXIÓN ENTRE PCs y SERVIDOR CON PARLANTES DEL SISTEMA DE MULTIDIFUSIÓN DE MENSAJES DE VOZ

Para la solución se ha tomado en cuenta las ventajas del Sistema Operativo Windows XP.

3.2.1 QUE SON LAS API

API es un término genérico que significa Aplication Programming Interface (Interface para la Programación de Aplicaciones). Sin embargo, en el contexto de la programación en Windows, esto se refiere específicamente a la API de Windows, la cual es el nivel mas bajo de interacción entre las aplicaciones y el sistema operativo. Los drivers, por su puesto, tienen aún niveles inferiores y trabajan con diferentes conjuntos de llamadas a funciones, pero para la gran mayoría de desarrollo en Windows esto no es relevante.

3.2.1.1 LA API DE WINDOWS

La API (Application Programming Interface) de Windows es una Interfaz para la programación de aplicaciones. Es en lo que se basan lo programadores para hacer compatible un programa con el sistema operativo. DirectX, por ejemplo, es un conjunto de APIs creada por Microsoft para mejorar el funcionamiento de aplicaciones multimedia y juegos en Windows 95/98/2000. Las funciones DirectX pueden ser utilizadas libremente por los programadores para mejorar sus productos. Las APIs encapsulan un conjunto de funciones propias del Sistema Operativo.

Todas estas funciones de Windows están escondidas dentro de unas cuantas DLLs ("Dynamic Link Library" o "Librería de Enlace Dinámico") agrupadas según la función que realizan:

- Kernel32.dll – operaciones de archivos y gestión de memoria.
- Gdi32.dll – operaciones gráficas
- User32.dll – maneja la parte de la interacción con el usuario.

También existen otros DLLs que, aunque menos importantes, se utilizaron en la presente investigación:

- Winmm.dll – se encarga de la parte multimedia
- Comdlg32.dll controles comunes para todas las aplicaciones

3.2.1.1.1 DLL (Dynamic Link Library)

Un DLL es una "Biblioteca de vínculos dinámicos" encapsulada en un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras DLLs. Se utilizan las DLLs para poder reciclar el código y aislar las diferentes tareas. Las DLLs no pueden ejecutarse directamente, es necesario llamarlas desde un código externo. Las DLLs de Windows permiten que las aplicaciones puedan operar en el entorno de Windows.

Normalmente las Bibliotecas de Vínculos Dinámicos aparecen con la extención **".dll"**; sin embargo, ellas también pueden tener la extensión **".exe"** u otra extensión. Por ejemplo, Shell.dll contiene las rutinas de arrastrar y soltar de **OLE** - ("Object Linking and Embedding" - *Incrustación y Vinculación de Objetos*) que pueden ser utilizadas por Windows y otros programas.

Kernel.exe, ***User.exe*** y ***Gdi.exe*** son un ejemplo de las DLLs con la extensión **.exe**. Estos archivos contienen código, datos o rutinas que se ejecutan en Windows. Por ejemplo, uno de estos archivos contiene la función **"CreateWindow"** que se utiliza cuando un programa quiere crear una nueva ventana en la pantalla.

En Windows un controlador que se instala es también una DLL. Un programa puede abrir, habilitar, consultar, deshabilitar y cerrar un controlador basado en las instrucciones escritas en un archivo DLL.

Las DLLs se pueden encontrar en el directorio de Windows, en el directorio Windows\System o en el directorio "Archivos de programa".

Si al iniciar un programa una DLL falta o está dañada, puede aparecer un mensaje de error como éste: "No se encuentra *xyz.dll*". Si la DLL de este programa está caduca o no corresponde al estándar se puede recibir el error "Llamada al vínculo dinámico no definido" ("Call to undefined dynalink").

En estos casos se debe obtener la DLL apropiada y colocarla en el directorio que corresponda.

3.2.1.1.1.1 REGISTRO DE UNA DLL

Algunos de los errores que presenta Windows se deben a que alguna DLLs no está debidamente registrada. Sobre todo esto suele suceder con lo que se refiere al acceso a datos (RDO350.DLL por ejemplo) y con los controles ActiveX (éstos no son las DLLs, son los ficheros con la extensión .OCX).

En este caso a veces puede funcionar registrar estos ficheros manualmente utilizando Regsvr32. El uso es

Regsvr32 [/u] [/s] <nombre del fichero>

Por ejemplo:

REGSVR32 c:\windows\system\Dao350.dll

Los parámetros opcionales [/u] [/s] significan lo siguiente:

[/u] - Para "desregistrar" una DLL (o un .ocx en vez de registrarlo).

[/s] - modo "silencioso" - no despliega los mensajes durante la operación

Los **OCXs** son objetos que haciendo uso de las capacidades del OLE permiten ampliar los elementos que se pueden incluir en los Formularios (forms) de Visual Basic.

3.2.1.1.1.2 CREACIÓN DE DLLS EN VISUAL BASIC

Desde VB sólo se puede crear un tipo de DLL: las DLL ActiveX. Estas DLL exportan objetos COM (ActiveX) que pueden ser automatizados por otras aplicaciones, pero no tienen algunas características de las DLL clásicas que puede escribir en C/C++ o en Delphi. Por ejemplo no puede exportar puntos de entrada a funciones. Al no poder exportar funciones sueltas con el método de llamada tradicional las DLL que se crean sólo pueden usarse desde otros lenguajes mediante automatización COM. Esto no es malo del todo ya que todos los lenguajes para Windows (incluyendo los de Script) soportan este tipo de automatización y podrán usar los recursos de una DLL. De hecho VB es uno de los lenguajes más utilizados para crear *DLLs de servidor en aplicaciones de Internet.*

Para crear una DLL en Visual Basic se escoge un tipo de proyecto especial en el menú que aparece cuando se ejecuta VB, o se puede cambiar simplemente el tipo de proyecto en las propiedades de éste. Para poder exportar objetos se debe aprender a crear clases y un poco de programación orientada a objetos ya que como se ha comentado con VB no se exporta funciones sino objetos COM automatizables.

Las DLLs que exportan funciones con convención de llamada para C y que están creadas en otros lenguajes se usan escribiendo la declaración de la función en un módulo, al igual que ocurre con las funciones de la API de Windows.

Los **módulos** son una de las formas que tiene Visual Basic de reutilizar código, un módulo es un archivo con extensión ".bas" añadido al proyecto como módulo. Todas las variables, funciones y subrutinas públicas podrán ser utilizadas desde el exterior del módulo, es decir desde el resto del proyecto, y las privadas únicamente dentro del módulo.

Para hacer referencia a elementos del módulo como variables, funciones y subrutinas se utiliza la notación: **modulo.elemento**, por ejemplo:

```
Modulo1.Variable1 = Valor
Variable1 = Modulo1.Funcion1( Parametro)
Modulo1.Rutina Parametro
```

A continuación se muestra un extracto de un módulo utilizado en el Sistema de Multidifusión de Voz. En el que se muestra la invocación de una función de una librería DLL en forma privada y la declaración de una función pública que podrá ser invocada desde cualquier parte del programa u otro módulo.

```
' Este código demuestra el uso de un módulo que invoca a la función PlaySound
' encapsulada dentro de la librería winm.dll

Module Sound  ' Función privada para uso interno del módulo
Private Declare Auto Function PlaySound Lib "winmm.dll"
    (ByVal lpszSoundName As String, ByVal hModule As Integer,  ByVal dwFlags As Integer) As Integer

' Función pública PlayWave que puede ser invocada desde cualquier otro módulo o código de programa
    Public Sub PlayWave()
```

```
        Dim fileName As String
        Const SND_FILENAME As Integer = &H20000
        fileName = System.IO.Path.GetDirectoryName(Application.ExecutablePath) &
"\MyWave.wav"
        PlaySound(fileName, 0, SND_FILENAME)
    End Sub
End Module
```

Un **Proyecto** de Visual Basic es un conjunto de todos los *módulos* y ficheros de los que consta una aplicación y los **Módulos** son zonas de la aplicación que contienen código. Cada modulo puede contener: Declaraciones y Procedimientos. En Visual Basic existen tres tipos de módulos:

1) de formulario (archivo *.frm)

2) estándar (archivo *.bas)

3) de clase (archivo *.cls)

1) Módulos de formulario (*.frm): Este tipo de módulos consta de los siguientes elementos:

- Un *formulario* con sus propiedades
- La información referente a los *controles* (y a sus propiedades) que contiene:
    i. El código programado en los *eventos* de esos controles y
    ii. Las *funciones* y *procedimientos* propios de ese formulario.
    iii. Código específico de la aplicación.

2) Módulos estándar (*.bas): Contienen declaraciones, procedimientos y funciones accesibles desde los otros módulos de la aplicación. El código no está ligado necesariamente a una aplicación determinada.

3) Módulos de clase (*.cls): Se usan para definir agrupaciones de datos y métodos llamadas clases. Son la base de la programación orientada a objetos.

Con toda la información anterior se puede resumir la creación de una DLL en visual Basic en los siguientes puntos:

1. Ejecutar Visual Basic y escoger un tipo especial de proyecto (DLL ActiveX), tal como se muestra en la Figura 7

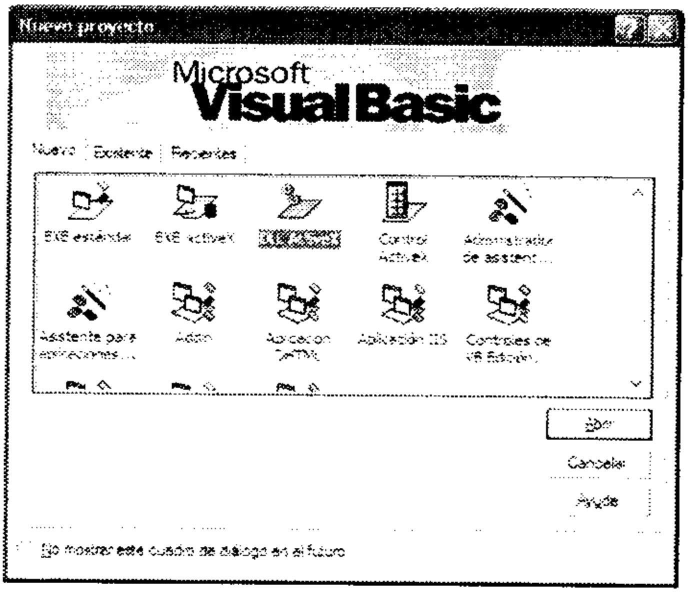


FIGURA 20. Selección de un tipo especial de Proyecto en Visual Basic

2. Realizar las declaraciones públicas o privadas, dentro del módulo de clase que se crea automáticamente al seleccionar lo indicado en la opción 1, o bien se puede crear nuevos módulos.
3. Una vez se haya terminado la codificación se puede genera la librería DLL, desde el menú Archivo-> Generar librería.dll
4. Escoger la ubicación física de éste nuevo dll y Aceptar (Al momento de generar la librería es necesario añadir la extensión .dll, caso contrario la aplicación no la pondrá.

3.2.2 PUERTO PARALELO

Los puertos son el medio para que las PCs se comuniquen con el mundo exterior. El nombre de puertos se debe a que cumplen con una función similar a los puertos de barcos. En éstos, los barcos pueden cargar y descargar productos, mientras que en los puertos de entrada y salida se posibilita la transmisión de información entre las PC y cualquier dispositivo externo. En la Figura 21. se muestra la información del Puerto paralelo.

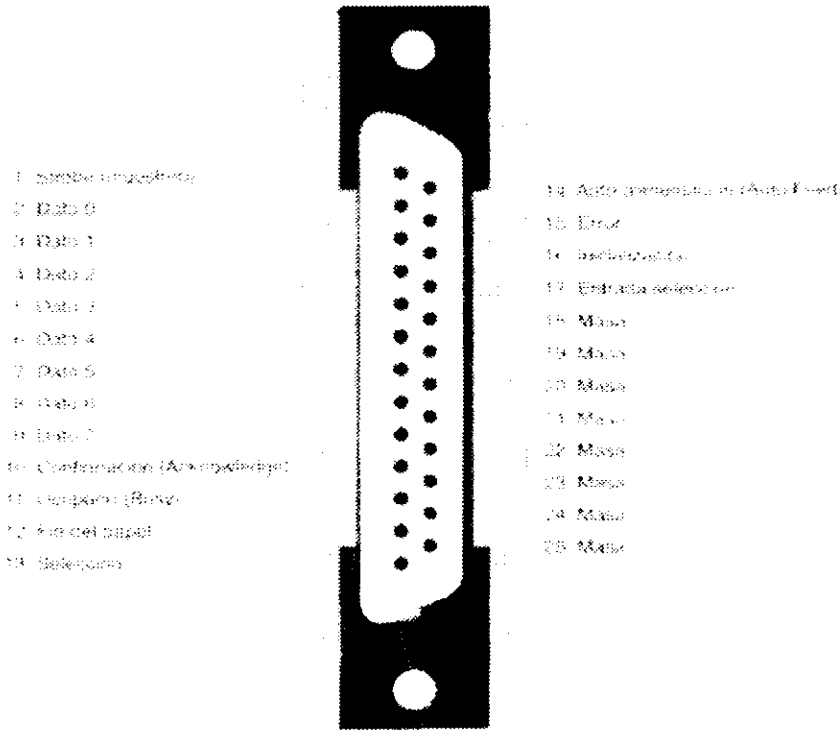


FIGURA 21. Información del Puerto Paralelo

En el Sistema SMDV (Sistema de Multidifusión de Voz) se están utilizando los pines del 2 al 9, que son las salidas de datos D0…D7 y el pin 25 como Tierra. En la Figura 22 se muestra la clasificación de los pines del puerto paralelo, es decir pines de entrada, salida, control y tierra.

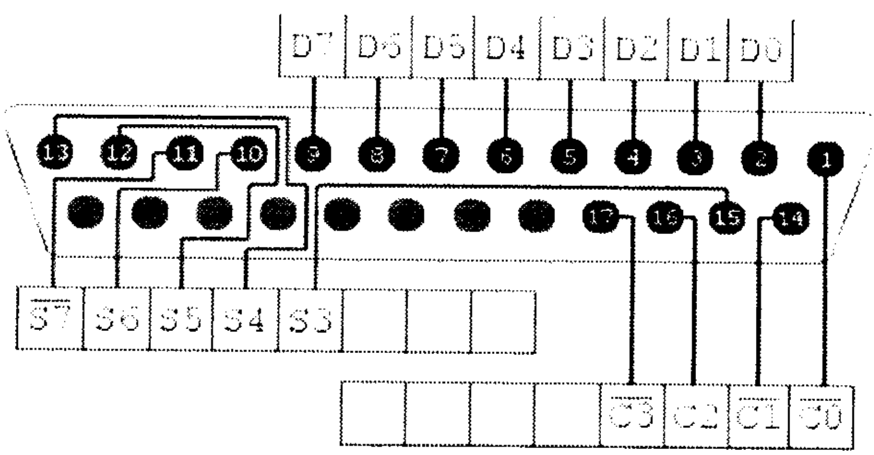


FIGURA 22. Clasificación de los pines del puerto paralelo

Para manipular el puerto paralelo desde el programa Servidor se utilizó un módulo llamada INPOUT32.BAS en el consta el siguiente código.

```
' Estas dos declaraciones utilizan la librería inpout32.dll para controlar el puerto paralelo
' Inp captura información enviada desde dispositivos externos
' Out escribe información en el puerto

Public Declare Function Inp Lib "inpout32.dll"
Alias "Inp32" (ByVal DirPuerto As Integer) As Integer

Public Declare Sub Out Lib "inpout32.dll"
Alias "Out32" (ByVal DirPuerto As Integer, ByVal Value As Integer)

' Declaración de una constante para guardar la dirección del puerto
' La dirección del puerto es un valor Hexadecimal que depende de cada computador.
' Los computadoes actuales utilizan la dirección &H378. Se puede verificar en
' el panel de control -> administrador de dispositivos
' Computadores antiguos utilizan &H3bc. Dependiendo de esto puede cambiar la
' siguiente variable:

Public Const DirPuerto   &H378
```

El código mostrado utiliza dos funciones existentes en la librería *inpout32.dll*, se realiza un alias de cada una de ellas, pero la que se utiliza en la aplicación es la función *Out*, por ejemplo para prender el parlante 1 se envía la instrucción Out Dirpuerto,1, para el parlante 2 Out Dirpuerto,2 y así sucesivamente.

3.2.3 CONEXIÓN EN RED

Para la comunicación entre el programa Servidor y el programa Cliente del SMDV se utilizan dos sockets (winsockets) que provee Visual Basic. El primero es utilizado para envío de información en texto y el segundo para envío de paquetes de sonido.

En el Servidor el primer socket se llama ***tcpServer*** y utiliza el puerto ***1544*** para comunicación con el Cliente en el cual se define el mismo puerto de comunicación. Dado que solo se utiliza un puerto de comunicación para envío/recepción, con éste socket únicamente se pueden establecer una conexión por vez. A través de éste socket el Servidor le envía la configuración de parlantes cuando un cliente se conecta, y por otro lado el Cliente le envía el parlante seleccionado de la lista enviada.

El segundo socket tanto en el Cliente como en el Servidor se llama ***TCPsokcet(0)***, está configurado para que se creen un arreglo de sockets, por eso es que se pueden tener múltiples conexiones de voz a la vez. Este socket utiliza el puerto ***701*** para escuchar solicitudes y el puerto ***3263*** para enviar paquetes de audio. En la Figura 23 se esquematiza lo antes mencionado.

aplicaciones deben usar el heap functions. Sin embargo, las funciones globales son usadas todavía con el DDE, el clipboard functions y el OLE data objects.

```
HGLOBAL GlobalAlloc(
  UINT uFlags,
  SIZE_T dwBytes
);
```

PARAMETROS

uFlags (banderas u)

[in] Asignación de atributos de la memoria. Si el cero es específico, la ausencia es GMEM_FIXED. Este parámetro puede ser uno o más de los valores siguientes, excepto por las combinaciones incompatibles que son específicamente notadas.

Valor	**Significado**
GHND	Combina GMEM_MOVEABLE y GMEM_ZEROINIT.
GMEM_FIXED	Asigna memorias fijas. El retorno del valor es un indicador.
GMEM_MOVEABLE	Asigna memorias movibles. Los bloques de memorias nunca son movidos a una memoria física, pero pueden ser movidos dentro de la pila faltante. El retorno del valor es un manejador al objeto de la memoria. Para trasladar el manejador a un

	indicador se usa la función del GlobalLock Este valor no puede ser combinado con GMEM_FIXED.
GMEM_ZEROINIT	Inicia los contenidos de la memoria a cero.
GPTR	Combina GMEM_FIXED y GMEM_ZEROINIT.

Los valores siguientes son obsoletos, pero son suministrados por compatibilidad con 16-bit de Windows. Ellos son ignorados.

GMEM_DDESHARE
GMEM_DISCARDABLE
GMEM_LOWER
GMEM_NOCOMPACT
GMEM_NODISCARD
GMEM_NOT_BANKED
GMEM_NOTIFY
GMEM_SHARE

dwBytes

[in] Número de bytes para designar. Si éste parámetro es cero y los parámetros de *uFlags (*banderas u*)* especifican GMEM_MOVEABLE, la función regresa a manejador a un objeto de memoria que es señalado o descartado.

Retorno de valores

Si la función tiene éxito, el retorno del valor es un manejador al reciente objeto de memoria designado.

Si la función falla, el retorno del valor es NULO. Para tener una información prolongada acerca del error existente, llame a GetLastError.

Observaciones

Si la pila no contiene suficiente espacio libre para satisfacer el pedido, **GlobalAlloc** se vuelve NULO. Debido a que NULO es usado para indicar un error, la dirección virtual de cero nunca es designada. Esto es, por lo tanto, fácil de detectar el uso de un indicador NULO.

La memoria designada con esta función es garantizada de ser alienada en un límite de 8-byte. Toda la memoria es creada con el acceso de ejecutar; ninguna función especial es requerida para ejecutar el código generado dinámicamente (dynamically generated code).

Si ésta función tiene éxito, éste designará al menos la cantidad de memoria requerida. Si la cantidad real designada es más grande que la cantidad requerida, el proceso puede usar la cantidad completa. Para determinar el número real de bytes designados, use la función GlobalSize.

Para liberar la memoria use la función GlobalFree.

Windows Me/98/95: Los administradores de la pila son diseñados por bloques de memoria más pequeños que cuatro megabytes. Si se espera que tus bloques de memoria sean mayores que uno o dos megabytes, se puede evitar degradaciones significantes en la función usando el VirtualAlloc o VirtualAllocEx, en vez de la función.

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servidor: Requiere Windows Server 2003, Windows 2000 Server, or Windows NTServer.
Cabecera: Declarado en Winbase.h; incluye Windows.h.
Biblioteca: Use Kernel32.lib.

GlobalFree

La función de **GlobalFree** libera el objeto de memoria global especificado e invalida sus manejos.

Nota las funciones globales son más lentas que otros manejos de funciones de memoria y no provee tantas características como ellas. Por lo tanto, las nuevas aplicaciones deben usar el heap functions. Sin embargo, las funciones globales son usadas todavía con el DDE, el clipboard functions

HGLOBAL GlobalFree(

HGLOBAL *hMem*

);

Parametros

hMem

[in] Manejador el objeto de memoria global. Este manejador es regresado por cualquiera de estas funciones el GlobalAlloc o el GlobalReAlloc.

Returno de valores

Si la función tiene éxito, el retorno del valor es NULO.

Si la función falla, el retorno del valor es igual a un manejador del objeto de memoria global. Para tener una información prolongada acerca del error existente, llame a GetLastError.

Observaciones

Si el proceso examina o modifica la memoria después de que haya sido liberado, la corrupción de la pila podría ocurrir o una excepción al acceso de violación podría ser generada. (EXCEPTION_ACCESS_VIOLATION)

Si el parámetro de *hMem* es NULO, **GlobalFree** falla y el sistema genera una excepción al acceso de violación.

La función de **GlobalFree** librará a un objeto de memoria sellado. Un objeto de memoria sellado tiene un contador de cerradura mayor que cero. La función de GlobalLock sella un objeto de memoria global e incrementa el contador de la cerradura por uno. La función del GlobalUnlock abre la cerradura y disminuye el contador de la cerradura por uno. Para tener el contador de la cerradura de un objeto de memoria global use la función del GlobalFlags.

Si una aplicación está funcionando bajo una versión de algún programa que quite algún virus de un programa en el ordenador del sistema, **GlobalFree** publicará un mensaje que dirá que el objeto sellado está siendo liberado. Si se está quitando algún virus de la aplicación, **GlobalFree** entrará al punto de ruptura justo antes de liberar al objeto sellado. Esto permite verificar el comportamiento intencionado, luego continúa con la ejecución.

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servidor: Requiere Windows Server 2003, Windows 2000 Server, o Windows NTServer.
Cabecera: Declarado en Winbase.h; incluye Windows.h.
Biblioteca: Usa Kernel32.lib.

GlobalLock

La función de **GlobalLock** sella un objeto de memoria global y regresa un señalador al primer byte del bloque de memoria del objeto.

Nota las funciones globales son más lentas que otros manejos de funciones de memoria y no provee tantas características como ellas. Por lo tanto, las nuevas aplicaciones deben usar el heap functions. Sin embargo, las funciones globales son usadas todavía con el DDE, el clipboard functions

LPVOID GlobalLock(

HGLOBAL *hMem*

);

Parametros

hMem

[in] Manejador del objeto de memoria global. Este manejador es regresado por cualquiera de estas funciones GlobalAlloc o GlobalReAlloc.

Retorno de valores

Si la función tiene éxito, el retorno de valores es un señalador del primer byte del bloque de memoria.

Si la función falla, el retorno del valor es NULO. Para tener una información prolongada acerca del error existente, llame a GetLastError.

Observaciones

La estructura interna del data para cada objeto de memoria, incluye un conteo de cerradura que estará inicialmente en cero. Para objetos de memoria movibles, **GlobalLock** incrementa el conteo por uno, y la función GlobalUnlock disminuirá el conteo por uno. Para cada llamada que un proceso realice a **GlobalLock** por un objeto, a la larga éste debe llamar a **GlobalUnlock**. La memoria sellada no será movida ni descartada, al menos que el objeto de memoria sea reasignado por el uso de la función GlobalReAlloc. El bloque de memoria de un objeto de memoria sellado permanece sellado hasta que su conteo de cerradura esté disminuido en cero, en el tiempo en el que pueda ser movido o descartado.

Objetos de memoria designados con GMEM_FIXED siempre tendrán un conteo de cerradura de cero. Para estos objetos, el valor del señalador regresado es igual al valor del manejador especificado.

Si el bloque de memoria especificado ha sido descartado o si el bloque de memoria tiene un tamaño de cero-byte esta función regresará a NULO.

Los objetos descartados siempre tendrán un conteo de cerradura de cero.

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servidor: Requiere Windows Server 2003, Windows 2000 Server, o Windows NTServer.
Cabecera: Declarado en Winbase.h; incluye Windows.h.
Biblioteca: Usa Kernel32.lib.

GlobalUnlock

La función **GlobalUnlock** disminuye el conteo de cerradura asociado con un objeto de memoria que fue asignado con GMEM_MOVEABLE. Esta función no tiene efecto en los objetos de memoria designados con GMEM_FIXED.

Nota las funciones globales son más lentas que otros manejos de funciones de memoria y no provee tantas características como ellas. Por lo tanto, las nuevas aplicaciones deben usar el heap functions. Sin embargo, las funciones globales son usadas todavía con el DDE, el clipboard functions.

BOOL GlobalUnlock(

HGLOBAL *hMem*

);

Parametros
hMem

[in] Manejador de objeto de memoria global. Este manejador es regresado por cualquiera de estas funciones GlobalAlloc o GlobalReAlloc.

Retorno de valores

Si el objeto de memoria está todavía sellado despues de decretar el conteo de la cerradura, el retorno del valor es un valor no cero.

Si la función falla, el retorno del valor es cero. Para tener una información prolongada acerca del error existente, llame a GetLastError. Si **GetLastError** retorna en NO_ERROR, el objeto de memoria ya no estará sellado.

Observaciones

La estructura interna del data para cada objeto de memoria incluye un conteo de cerradura que estará inicialmente en cero. Para los objetos de memoria movibles, la función GlobalLock incrementará el conteo por uno y **GlobalUnlock** disminuirá el conteo por uno. Para cada llamada que un proceso hace a **GlobalLock** por un objeto, a la larga éste llamará a **GlobalUnlock**. La memoria sellada no será movida ni descartada a menos que el objeto de memoria sea reasignado por el uso de la función GlobalReAlloc. El bloque de memoria de un objeto de memoria sellado permanecerá sellado hasta que su conteo de cerradura sea disminuido a cero, tiempo en el cual éste podrá ser movido o descartado.

Los objetos de memoria asignados con GMEM_FIXED siempre tienen un conteo de cerradura de cero. Si el bloque de memoria especificado es una memoria fija, esta función regresa a VERDADERO.

Si el objeto de memoria ya no está sellado, **GlobalUnlock** regresa a FALSO y **GetLastError** reporta ERROR_NOT_LOCKED.

Un proceso no debe contar con el retorno de valor para determinar el número de veces que éste debe llamar subsiguientemente a **GlobalUnlock** por un objeto de memoria.

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servido: Requiere Windows Server 2003, Windows 2000 Server, o Windows NT Server.

Cabecera: Declarado in Winbase.h; incluye Windows.h.
Biblioteca: Use Kernel32.lib.

Funciones Multimedia

Las siguientes funciones son usadas con multimedia. ACM (Audio Compression Manager)

acmDriverAdd
acmDriverClose
acmDriverDetails
acmDriverEnum
acmDriverEnumCallback
acmDriverID
acmDriverMessage
acmDriverOpen
acmDriverPriority
acmDriverProc
acmDriverRemove
acmFilterChoose
acmFilterChooseHookProc
acmFilterDetails
acmFilterEnum
acmFilterEnumCallback
acmFilterTagDetails
acmFilterTagEnum
acmFilterTagEnumCallback
acmFormatChoose
acmFormatChooseHookProc
acmFormatDetails
acmFormatEnum
acmFormatEnumCallback
acmFormatSuggest
acmFormatTagDetails
acmFormatTagEnum
acmFormatTagEnumCallback
acmGetVersion
acmMetrics
acmStreamClose
acmStreamConvert
acmStreamConvertCallback
acmStreamMessage
acmStreamOpen
acmStreamPrepareHeader
acmStreamReset
acmStreamSize
acmStreamUnprepareHeader
auxGetDevCaps
auxGetNumDevs
auxGetVolume
auxOutMessage
auxSetVolume
AVIBuildFilter
AVIClearClipboard
AVIFileAddRef
AVIFileCreateStream
AVIFileEndRecord
AVIFileExit
AVIFileGetStream
AVIFileInfo
AVIFileInit
AVIFileOpen

AVIFileReadData
AVIFileRelease
AVIFileWriteData
AVIGetFromClipboard
AVIMakeCompressedStream
AVIMakeFileFromStreams
AVIMakeStreamFromClipboard
AVIPutFileOnClipboard
AVISave
AVISaveOptions
AVISaveOptionsFree
AVISaveV
AVIStreamAddRef
AVIStreamBeginStreaming
AVIStreamCreate
AVIStreamEndStreaming
AVIStreamFindSample
AVIStreamGetFrame
AVIStreamGetFrameClose
AVIStreamGetFrameOpen
AVIStreamInfo
AVIStreamLength
AVIStreamOpenFromFile
AVIStreamRead
AVIStreamReadData
AVIStreamReadFormat
AVIStreamReleaseAVIStreamS
ampleToTime
AVIStreamSetFormat
AVIStreamStart
AVIStreamTimeToSample
AVIStreamWrite
AVIStreamWriteData

capControlCallback
capCreateCaptureWindow
capErrorCallback
capGetDriverDescription
capStatusCallback
capVideoStreamCallback
capWaveStreamCallback
capYieldCallback
CreateEditableStream
DllGetClassObject
DrawDibBegin
DrawDibChangePalette
DrawDibClose
DrawDibDraw
DrawDibEnd
DrawDibGetBuffer
DrawDibGetPalette
DrawDibOpen
DrawDibProfileDisplay
DrawDibRealize
DrawDibSetPalette
DrawDibStart
DrawDibStop
DrawDibTime
EditStreamClone
EditStreamCopy
EditStreamCut
EditStreamPaste
EditStreamSetInfo
EditStreamSetName
GetOpenFileNamePreview
GetSaveFileNamePreview
ICClose

ICCompress
ICCompressorChoose
ICCompressorFree
ICDecompress
ICDecompressEx
ICDecompressExBegin
ICDecompressExQuery
ICDraw
ICDrawBegin
ICDrawSuggestFormat
ICGetDisplayFormat
ICGetInfo
ICImageCompress
ICImageDecompress
ICInfo
ICInstall
ICLocate
ICOpen
ICOpenFunction
ICRemove
ICSendMessage
ICSeqCompressFrame
ICSeqCompressFrameEnd
ICSeqCompressFrameStart
ICSetStatusProc
IOProc
joyConfigChanged
joyGetDevCaps
joyGetNumDevs
joyGetPos
joyGetPosEx
joyGetThreshold
joyReleaseCapture
joySetCapture
joySetThreshold
mciExecute
mciGetCreatorTask
mciGetDeviceID
mciGetDeviceIDFromElementI
D
mciGetErrorString
mciGetYieldProc
mciSendCommand
mciSendString
mciSetYieldProc
MCIWndCreate
MCIWndRegisterClass
midiConnect
midiDisconnect
midiInAddBuffer
midiInClose
midiInGetDevCaps
midiInGetErrorText
midiInGetID
midiInGetNumDevs
midiInMessage
midiInOpen
midiInPrepareHeader
MidiInProc
midiInReset
midiInStart
midiInStop
midiInUnprepareHeader
midiOutCacheDrumPatches
midiOutCachePatches
midiOutClose

midiOutGetDevCaps
midiOutGetErrorText
midiOutGetID
midiOutGetNumDevs
midiOutGetVolume
midiOutLongMsg
midiOutMessage
midiOutOpen
midiOutPrepareHeader
MidiOutProc
midiOutReset
midiOutSetVolume
midiOutShortMsg
midiOutUnprepareHeader
midiStreamClose
midiStreamOpen
midiStreamOut
midiStreamPause
midiStreamPosition
midiStreamProperty
midiStreamRestart
midiStreamStop
mixerClose
mixerGetControlDetails
mixerGetDevCaps
mixerGetID
mixerGetLineControls
mixerGetLineInfo
mixerGetNumDevs
mixerMessage
mixerOpen
mixerSetControlDetails
mmioAdvance
mmioAscend
mmioClose
mmioCreateChunk
mmioDescend
mmioFlush
mmioGetInfo
mmioInstallIOProc
mmioOpen
MMIOProc
mmioRead
mmioRename
mmioSeek
mmioSendMessage
mmioSetBuffer
mmioSetInfo
mmioStringToFOURCC
mmioWrite
MyStatusProc
PlaySound
sndPlaySound
StretchDIB
timeBeginPeriod
timeEndPeriod
timeGetDevCaps
timeGetSystemTime
timeGetTime
timeKillEvent
TimeProc
timeSetEvent
waveInAddBuffer
waveInClose
waveInGetDevCaps
waveInGetErrorText

waveInGetID
waveInGetNumDevs
waveInGetPosition
waveInMessage
waveInOpen
waveInPrepareHeader
waveInProc
waveInReset
waveInStart
waveInStop
waveInUnprepareHeader
waveOutBreakLoop
waveOutClose
waveOutGetDevCaps
waveOutGetErrorText
waveOutGetID
waveOutGetNumDevs
waveOutGetPitch
waveOutGetPlaybackRate
waveOutGetPosition
waveOutGetVolume
waveOutMessage
waveOutOpen
waveOutPause
waveOutPrepareHeader
waveOutProc
waveOutReset
waveOutRestart
waveOutSetPitch
waveOutSetPlaybackRate
waveOutSetVolume
waveOutUnprepareHeader
waveOutWrite

acmStreamOpen

La función **acmStreamOpen** abre una corriente de conversión ACM (Audio Compression Manager, Administrador de Compresion de Audio). Las corrientes de conversiones son usadas para transformarlos de un formato de audio específico a otro.

MMRESULT acmStreamOpen(

LPHACMSTREAM *phas*,

HACMDRIVER *had*,

LPWAVEFORMATEX *pwfxSrc*,

LPWAVEFORMATEX *pwfxDst*,

LPWAVEFILTER *pwfltr*,

DWORD_PTR *dwCallback,*

DWORD_PTR *dwInstance,*

DWORD *fdwOpen*

);

Parametros

phas

Señalador a manejador. Recibirán el nuevo manejador de corriente que puede ser usado para las funciones de las conversiones. Este manejador es usado para identificar la corriente en llamadas a otras funciones ACM conversiones de corrientes. Si la bandera ACM_STREAMOPENF_QUERY es especificada, este parámetro debe ser NULO.

had

Manejador a un conductor ACM. Si el manejador es especificado, éste identifica un conductor específico que puede ser usado por una conversión de corriente. Si este parámetro es NULO, todos los conductores ACM instalados adecuadamentes serán cuestionados hasta que un competidor o rival sea encontrado.

pwfxSrc

Señalador a un WAVEFORMATEX. Esrtructura que identifica el formato del recurso deseado para la conversion.

pwfxDst

Señalador a un **WAVEFORMATEX.** Estructura que identifica el formato de la destinación deseada para la conversión.

pwfltr

Parametros

has

Manejador a la corriente de conversión.

pash

Señalador a una estructura ACMSTREAMHEADER que identifica la fuente y amortiguadores destinados para ser preparados.

fdwPrepare

Reservado, debe ser cero.

Retorno de valores

Retorna a cero si es exitoso o de otro modo hay un error. Valores de posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALFLAG	Al menos una bandera es inválida.
MMSYSERR_INVALHANDLE	El manejador especificado es inválido.
MMSYSERR_INVALPARAM	Al menos un parámetro es inválido.
MMSYSERR_NOMEM	El sistema es incapás de designer recursos.

Observaciones

Preparar un manejador de corriente que ya ha sido preparado no tiene efecto, y las funciones regresan a cero. Sin embargo se debe asegurar que las aplicaciones no prepararen un manejador de corriente en múltiples veces.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después

Cabecera: Declarado en Msacm.h.

Biblioteca: Usa Msacm32.lib.

acmStreamUnprepareHeader

La función **acmStreamUnprepareHeader** limpia la preparación ejecutada por la función acmStreamPrepareHeader para una corriente ACM. Esta función debe ser llamada, después el ACM es terminado con los amortiguadores dados. Una aplicación debe llamar a esta función antes de liberar la fuente y los amortiguadores destinados.

MMRESULT acmStreamUnprepareHeader(

HACMSTREAM *has*,

LPACMSTREAMHEADER *pash*,

DWORD *fdwUnprepare*

);

Parametros

has

Manejador a la corriente de conversión.

pash

Señalador a la estructura ACMSTREAMHEADER que identifica la fuente y los amortiguadores destinados para no ser preparados.

fdwUnprepare

Reservado, debe ser cero.

Retorno de valores

acmStreamUnprepareHeader regresa con éxito, el manejador siempre sera no preparado.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después
Windows 95/98/Me: Incluido en Windows 95 y después
Cabecera: Declarado en Msacm.h.
Bblioteca: Usa Msacm32.lib.

acmStreamConvert

La función **acmStreamConvert** require el ACM para ejecutar una transformación en una corriente de conversión específica. Una transformación puede ser sincronizada o no, dependiendo en cómo la corriente fuera abierta.

MMRESULT acmStreamConvert(

HACMSTREAM *has*,

LPACMSTREAMHEADER *pash*,

DWORD *fdwConvert*

);

Parametros

has

Manejador a la corriente de conversión abierta.

pash

Señalador a un manejador de corriente que describa la fuente y los amortiguadores destinados para una transformación. Este manejador debe haber sido preparado previamente para usar la función acmStreamPrepareHeader.

fdwConvert

Banderas para hacer la transformación. Los valores siguientes están definidos.

Valor	Significado
ACM_STREAMCONVERTF_BLOCKALIGN	Solo números integrales de bloques serán transformados. Convirtiendo la información terminará en límites de bloques alineados. Una aplicación debe usar esta bandera para todas las transformaciones en una corriente hasta que no haya suficiente fuente de información para transformar a un destinode bloques alineados. En este caso, la última transformación debe ser especificada sin esta bandera.
ACM_STREAMCONVERTF_END	La corriente de conversion ACM debe empezar regresando hasta la instancia de la información. Por ejemplo si una corriente de conversión sostiene la instancia de la información, como el fin del eco de la operación de filtro, esta bandera causará la corriente para empezar a regresar esta información restante con fuente de información opcional. Esta bandera puede ser especificada con la bandera ACM_STREAMCONVERTF_START
ACM_STREAMCONVERTF_START	La corriente de conversion ACM debe reiniciar la instancia de la información. por ejemplo si una corriente de conversión sostiene la instancia del

Windows NT/2000: La precisión faltante de la función **timeGetTime** puede ser de 5 milisegundos o más, dependiendo de la máquina. Usted puedes usar las funciones timeBeginPeriod y timeEndPeriod para incrementar la precisión de **timeGetTime**. Se hace eso, la diferencia minima entre valores sucesivos regresan por **timeGetTime** puede ser tan larga como el valor del período mínimo puesto usando **timeBeginPeriod** y **timeEndPeriod**. usa las funciones **QueryPerformanceCounter** y **QueryPerformanceFrequency** para medir pequeños intervalos de tiempo con una alta resolución.

Windows 95: La precisión faltante en la función **timeGetTime** es de 1 milisegundo. En otras palabras, la función **timeGetTime** puede regresar valores sucesivos que difieren justo por 1 milisegundo. Esto es verdad, sin importar que llamadas han sido hechas a las funciones **timeBeginPeriod** y **timeEndPeriod**.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después
Windows 95/98/Me: Incluido enWindows 95 y después
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

sndPlaySound

La función **sndPlaySound** ejecuta una forma de onda de audio especificada por un archivo de nombres o por una entrada en el registro o por el archivo WIN.INI. Esta función ofrece un sub puesto de funcionalidad de la función PlaySound; **sndPlaySound** está siendo mantenida por compatibilidad hacia atrás.

BOOL sndPlaySound(

LPCSTR *lpszSound*,

UINT *fuSound*

);

Parametros

lpszSound

Una cuerda que especifica el sonido para ejecutar. Este parámetro puede ser una entrada al registro o un WIN.INI que identifica un sistema de sonido, o puede ser el nombre de una forma de onda de un archivo de audio. (Si la función no encuentra la entrada, este parámetro es asistido como un archivo de nombres). Si el parámetro es NULO, ningun sonido actualmente actuando es parado.

fuSound

Banderas para enviar el sonido. Los valores siguientes están definidos.

Valor	**Significado**
SND_ASYNC	El sonido es ejecutado sin sincronizar y la función retorna inmediatamente luego de empezar el sonido. Para terminar un sonido enviado sin sincronía, llama **sndPlaySound** con *lpszSoundName* puesto a NULO.
SND_LOOP	El sonido ejecuta repetidamente hasta que **sndPlaySound** es llamado nuevamente con el parámetro *lpszSoundName* puesto a NULO. También se debe especificar la bandera SND_ASYNC para sujetar sonidos.
SND_MEMORY	Este parámetro especificado por *lpszSoundName* señala a una imagen de una forma de onda de sonido en una memoria.
SND_NODEFAULT	Si el sonido no puede ser encontrado, la función regresa silenciosamente sin actuar el sonido faltante.
SND_NOSTOP	Si el sonido esta actualmente sonando, la función regresa inmediatamente a FALSO, sin sonar el sonido pedido.

SND_SYNC	El sonido es actuado sincronizadamente y la función no regresa hasta que el sonido termine.

Retorno de valores

Regresa verdaderdo si tiene éxito, de otro mos es FALSO.

Observaciones

Si el sonido especificado no puede ser encontrado, **sndPlaySound** pondrá el sistema de sonido faltante. Si no hay un sistema faltante entrando en el registro o en el archivo de WIN.INI, o si la falta de sonido no puede ser encontrada, la función no hace ningún sonido y regresa a FALSO.

El sonido especificado debe ajustar una memoria física disponible y ser ejecutado por un conductor instalado de forma de onda del dispositivo de audio. Si **sndPlaySound** no encuentra el sonido en un directorio actual, la función lo busca por el uso de una orden en el buscador de directorio estandar.

Winmm.dll exporta **sndPlaySoundA** para el carácter puesto ANSI y **sndPlaySoundW** para caracteres puestos de Unicode. Cual función es realmente llamada depende en si el código es definido en la cabezera para recopilar ANSI o Unicode.

Windows 95/98/Me: sndPlaySoundW es apoyado por Microsoft Layer for Unicode. Para usarlo, se debe añadir archivos seguros para la aplicación, como perfilado en Microsoft Layer for Unicode on Windows 95/98/Me Systems.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después
Windows 95/98/Me: Incluido in Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.
Unicode: Implementado como un Unicode y ANSI versions en Windows NT/2000/XP. También apoyadas por Microsoft Layer for Unicode.

LPWAVEHDR *pwh*,

UINT *cbwh*

);

Parametros

hwi

Manejador al dispositivo de la información de la forma de onda del audio (the waveform-audio).

pwh

Señalador de una estructura WAVEHDR que identifica el amortiguador para ser preparado.

cbwh

Tamaño en bytes, de la estructura **WAVEHDR**.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito, caso contrario si tiene un error. Los valores de los posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Inválido reserva o bloquear memoria.

Observaciones

Los miembros **lpData**, **dwBufferLength**, y **dwFlags** de la estructura **WAVEHDR** debe ser puesto antes de llamar a su función (**dwFlags** debe ser cero).

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveInUnprepareHeader

La función **waveInUnprepareHeader** limpia la preparación ejecutada por la función waveInPrepareHeader. Esta función debe ser llamada después de que el conductor del dispositivo llene un amortiguador y lo retorne a la aplicación. Se debe llamar a esta función antes de liberar al amortiguador.

MMRESULT waveInUnprepareHeader(

HWAVEIN ***hwi*,**

LPWAVEHDR ***pwh*,**

UINT ***cbwh***

);

Parametros

hwi

Llame a esta función cuando la información puesta ya haya empezado y no tenga efecto, y la función retornará a cero.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabezera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveInStop

La función **waveInStop** para la información puesta en la forma de onda del audio.

MMRESULT waveInStop(

HWAVEIN *hwi*

);

Parametros

hwi

Manejador del dispositivo de información de la forma de onda del audio.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si hay algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está

	presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.

Observaciones

Si hay algún amortiguador en la fila de espera, el amortiguador actual sera marcado como realizado (el miembro **dwBytesRecorded** en la cabecera tendrá la longitud de la información), pero ningún amortiguador vacío en la fila de espera se mantendrá ahí.

Usted debe llamar a esta función cuando la puesta de información no esté iniciada y no tenga efecto, y la función regresará a cero.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después
Windows 95/98/Me: Incluido en Windows 95 y después
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveInReset

La función **waveInReset** para la puesta de información en el dispositivo dado de información puesta de la forma de onda de audio y reajuste la posición actual a cero. Todos los amortiguadores pendientes son marcados como realizados y regresados a la aplicación.

MMRESULT waveInReset(

HWAVEIN *hwi*

);

Parametros

hwi

Manejador del dispositivo de información de la forma de onda del audio.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si tiene algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	Descripción
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y despues
Windows 95/98/Me: Incluido en Windows 95 y despues.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveInClose

La función **waveInClose** cierra el manejador del dispositivo dado de información de la forma de onda del audio.

MMRESULT waveInClose(

HWAVEIN *hwi*

);

Parametros

hwi

Manejador del dispositivo de información de la forma de onda del audio. Si la función tiene éxito, ya no será más válido el manejador después de esta llamada.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si tiene algún error, los valores de los posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar memorias.
WAVERR_STILLPLAYING	Todavía hay amortiguadores en la fila de espera.

Observaciones

Si hay amortiguadores con información puesta que han sido enviados con la función **waveInAddBuffer** y no ha sido regresada a la aplicación, la operación de cerrar fallará. Llame a la función **waveInReset** para señalar todos los amortiguadores pendientes como realizados.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutOpen

La función **waveOutOpen** abre el dispositivo previo al saliente de la forma de onda de audio.

MMRESULT waveOutOpen(

LPHWAVEOUT ***phwo*,**

UINT_PTR ***uDeviceID*,**

LPWAVEFORMATEX ***pwfx*,**

DWORD_PTR ***dwCallback*,**

DWORD_PTR ***dwCallbackInstance*,**

DWORD *fdwOpen*

);

Parametros

phwo

Señalador a un amortiguador que recibe un manejador que identifica el dispositivo abierto saliente de la forma de onda de audio. Se usa el manejador para identificar el dispositivo cuando se llama a otras funciones salientes de forma de onda de audio. Este parámetro podría ser NULO si la bandera WAVE_FORMAT_QUERY es especificada por *fdwOpen*.

uDeviceID

Identificador del dispositivo saliente de la forma de onda de audio para abrir. Puede ser un dispositivo identificador o un manejador de un dispositivo entrante abierto de la forma de onda de audio. Se puede usar la siguiente bandera ejemplar de un dispositivo identificador.

Valor	**Significado**
WAVE_MAPPER	La función selecciona un dispositivo saliente de foma de onda de audio capás de ejecutar el formato dado.

pwfx

Señalador a una estructura WAVEFORMATEX que identifica el formato del data de la forma de onda de audio para ser enviado al dispositivo. Puedes liberar esta estructura inmediatamente después de que pase al **waveOutOpen**.

dwCallback

Señalador a una función fija callback, un manejador de evento, un manejador a una ventana, o un identificador de un cabo para ser llamado durante la previa forma de onda de audio para procesar mensajes relacionados al progreso de la previa grabación de sonido. Si ninguna función de callback es requerida, este valor puede ser cero. Para más información de la función callback, mire waveOutProc.

dwCallbackInstance

Usador ejemplar del data pasado al mecanismo de callback. Este parámetro no es usado con la ventana del mecanismo de callback.

fdwOpen

Banderas para abrir el dispositivo. Los valores siguientes están definidos.

Valor	**Significado**
CALLBACK_EVENT	El parámetro *dwCallback* es un manejador de evento.
CALLBACK_FUNCTION	El parámetro *dwCallback* es una dirección de procedimiento de callback.

CALLBACK_NULL	No hay mecanismo de callback mechanism. Esta es la versión faltante.
CALLBACK_THREAD	El parámetro *dwCallback* es un identificador de cabo.
CALLBACK_WINDOW	El parámetro *dwCallback* es un manejador de ventana.
WAVE_ALLOWSYNC	Si la bandera es especificada, un dispositivo sincronizado de la forma de onda de audio puede ser abierto. Si la bandera no es especificada mientras se abre un conductor sincronizado, el dispositivo fallará la apertura.
WAVE_FORMAT_DIRECT	Si la bandera es especificada, el conductor ACM no ejecutará transformaciones en el data de audio.
WAVE_FORMAT_QUERY	Si la bandera es especificada, **waveOutOpen** cuestionará el dispositivo para determinar si sus apoyos dan el formato, pero el dispositivo no es realmente abierto.
WAVE_MAPPED	Si la bandera es especificada, el parámetro *uDeviceID* especifica un dispositivo de la forma de onda de audio para ser graficado por la onda graficadora.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si hay un error. Los valores de los posibles errores incluyen lo siguiente.

Valor	Descripción
MMSYSERR_ALLOCATED	El recurso especificado ya es asignado.
MMSYSERR_BADDEVICEID	El dispositivo identificador especificado esta fuera de alcance.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.
WAVERR_BADFORMAT	Intentó abrir con un formato no apoyado de la forma de onda de audio.
WAVERR_SYNC	El aparato es sincronizado pero **waveOutOpen** fue llamado sin usar la bandera WAVE_ALLOWSYNC.

Observaciones

Use la función waveOutGetNumDevs para determinar el número de dispositivos salientes de la forma de onda de audio presente en el sistema. Si el valor especificado por el parámetro *uDeviceID* es un dispositivo identificador, puede variar desde cero hasta uno menos que el número de dispositivos presentes. La constante WAVE_MAPPER puede también ser usada como un dispositivo identificador.

La estructura señalada por *pwfx* puede ser extendida para incluir información de cintas específicas para seguros formatos de data. Por ejemplo para un data PCM data, un extra **UINT** es añadida para especificar el número de bits por muestra. Se usa la estructura PCMWAVEFORMAT en este caso. Para todos los otros formatos de la forma de onda de audio, use la estructura **WAVEFORMATEX** para especificar la longitude de la información adicional.

Si se escoge tener una ventana o un cabo para recibir información de callback, los mensajes siguientes son enviados a la ventana de función de procedimientos

Cabecera: Declarado en Mmsystem.h; incluye Windows.h. **Biblioteca:** Use Winmm.lib.

waveOutUnprepareHeader

La función **waveOutUnprepareHeader** limpia la preparación ejecuta por la función waveOutPrepareHeader. Esta función debe ser llamada después de que el conductor de dispositivo esté terminado con un bloque de data. Se debe llamar a esta función antes de liberar al amortiguador.

MMRESULT waveOutUnprepareHeader(

HWAVEOUT *hwo***,**

LPWAVEHDR *pwh***,**

UINT *cbwh*

);

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio.

pwh

Señalador de una estructura WAVEHDR que identifica el bloque de data para ser limpiado.

cbwh

Tamaño en bytes, de la estructura **WAVEHDR**

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito, de otro modo si tiene un error. Los valores de los errors siguientes incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.
WAVERR_STILLPLAYING	El bloque de data señalado por el parámetro *pwh* está todavía en la fila de espera.

Observaciones

Esta función complementa **waveOutPrepareHeader**. Se debe llamar a esta función antes de liberar al amortiguador. Después de pasar un amortiguador al conductor del dispositivo con la función **waveOutWrite,** se debe esperar hasta que el conductos esté terminado con el amortiguador antes de llamar a **waveOutUnprepareHeader**.

No preparar un amortiguador que no ha sido preparado, no tiene efecto, y la función regresa a cero.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutWrite

La función **waveOutWrite** envía un bloque de data al dispositivo dado saliente de la forma de onda de audio.

```
MMRESULT waveOutWrite(
  HWAVEOUT hwo,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio.

pwh

Señalador a una estructura WAVEHDR contiene información acerca del bloque de data.

cbwh

Tamaño en bytes, de la estructura **WAVEHDR**.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito, o de otro modo si tiene algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	Descripción
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es

	inválido.
MMSYSERR_NODRIVER	Ningún conductor del dispositivo está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.
WAVERR_UNPREPARED	El bloque de data señalado por el parámetro *pwh* no ha sido preparado.

Observaciones

Cuando el amortiguador esté terminado, el bite WHDR_DONE es puesto en el miembro **dwFlags** de la estructura **WAVEHDR**.

El amortiguador debe ser preparado con la función **waveOutPrepareHeader** antes de que éste sea pasado a **waveOutWrite**. A menos que el dispositivo esté pausado por la llamada de la función **waveOutPause,** la grabación previa empieza cuando el primer bloque de dataes enviado al dispositivo.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutClose

La función **waveOutClose** cierra el dispositivo saliente de la forma de onda de audio.

```
MMRESULT waveOutClose(
  HWAVEOUT hwo
);
```

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio. Si la función tiene éxito, el manejador ya no es más válido después de ésta llamada.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si hay algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	Descripción
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.
WAVERR_STILLPLAYING	Todavía hay amortiguadores en la fila de espera.

Observaciones

Si el dispositivo todavía está ejecutando un archivo de la forma de onda del audio, la operación para cerrar falla. Se usa la función **waveOutReset** para terminar la grabación previa antes de llamar **waveOutClose**.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después
Windows 95/98/Me: Incluido en Windows 95 y después
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutReset

La función **waveOutReset** para la grabación previa en el dispositivo dado saliente de la forma de onda de audio y reajusta la posición actual a cero. Todos los amortiguadores de grabaciones previas pendientes son marcados como realizados y regresados a la aplicación.

MMRESULT waveOutReset(

HWAVEOUT *hwo*

);

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio.

Retorno de Valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si tiene hay algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	No puede sellar la memoria.
MMSYSERR_NOTSUPPORTED	El dispositivo especificado está sincronizado y no apoya la pausa.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutPause

La función **waveOutPause** pausa la grabación previa en el dispositivo dado saliente de la forma de onda de audio. La posición actual es guardada. Se usa la función waveOutRestart para reaunudar la grabación previa desde la posición actual.

MMRESULT waveOutPause(

HWAVEOUT *hwo*

);

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio.

Retorno de valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si hay algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	**Descripción**
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.

MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	No puede sellar la memoria.
MMSYSERR_NOTSUPPORTED	El dispositivo especificado está sincronizado y no apoya la pausa.

Observaciones

Se llama a esta función cuando la saliente ya esté pausada y no tenga efecto, y la función regrese a cero.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

waveOutRestart

La función **waveOutRestart** reanuda la grabación previa en un dispositivo saliente pausado de la forma de onda de audio.

MMRESULT waveOutRestart(

HWAVEOUT *hwo*

);

Parametros

hwo

Manejador del dispositivo saliente de la forma de onda del audio.

Retorno de Valores

Retorna MMSYSERR_NOERROR si tiene éxito o de otro modo si hay algún error. Los valores de los posibles errores incluyen lo siguiente.

Valor	Descripción
MMSYSERR_INVALHANDLE	El manejador del dispositivo especificado es inválido.
MMSYSERR_NODRIVER	Ningún conductor de dispositivos está presente.
MMSYSERR_NOMEM	Incapaz de asignar o sellar la memoria.
MMSYSERR_NOTSUPPORTED	El aparato especificado está sincronizado y no apoya la pausa.

Observaciones

Llame a esta función cuando la saliente no esté pausada y no tenga efecto, y la función regresa a cero.

Requisitos

Windows NT/2000/XP: Incluido en Windows NT 3.1 y después.
Windows 95/98/Me: Incluido en Windows 95 y después.
Cabecera: Declarado en Mmsystem.h; incluye Windows.h.
Biblioteca: Use Winmm.lib.

CopyMemory

La función **CopyMemory** copia un bloque de memoria desde una localización a otra.

void CopyMemory(

PVOID *Destination*,

const VOID* *Source*,

SIZE_T *Length*

);

Parametros

Destination

[in] Señalador a la primera dirección de la desrincaión del bloque copiado.

Source

[in] Señalador a la primera dirección del bloque de memoria para copiar.

Length

[in] Tamaño del bloque de memoria para copiar en bytes.

Retorno de valores

Esta función no tiene retorno de valores.

Observaciones

Esta función es definida como la función **RtlCopyMemory.** Para más información mire Winbase.h y Winnt.h.

Si la fuente y y el destino de los bloques coinciden, los resultados no son definidos. Para bloques coincidos, use la función **MoveMemory**.

Advertencia El primer parámetro, *Destination*, debe ser lo suficientemente largo para sostener *Length* bytes de *Source*; de otro modo, un amortiguador infector podría ocurrir. Este puede guiar a una negativa del servicio de ataque contra la aplicación si un acceso de violación ocurre, o, en el peor de los casos, si designa un ataque para inyectar un códico ejecutable en el proceso. Esto es especialmente verdad si *Destination* es un amortiguador amontonado en parte inferior (stack-based buffer). Sea conciente de que el ultimo parámetro, *Length*, es el número de bytes para copiar dentro de *Destination*, y no el tamaño de *Destination*.

Los siguientes ejemplos de códigos muestra una vía segura para usar **CopyMemory**:

```
void test(char *pbData, unsigned int cbData) {

  char buf[BUFFER_SIZE];

  CopyMemory(buf, pbData, min(cbData,BUFFER_SIZE));

}
```

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servidor: Requiere Windows Server 2003, Windows 2000 Server, o Windows NT Server.
Cabecera: Declarado en Winbase.h; incluye Windows.h.

IsBadCodePtr

Esta función **IsBadCodePtr** determina si el proceso de llamar ha leido el acceso a la memoria de la dirección especificada.

BOOL IsBadCodePtr(

  FARPROC *lpfn*

);

Parametros

lpfn

[in] Señalador a la dirección de memoria.

Retorno de Valores

Si el proceso de llamada ha leido el acceso a la memoria especificada, el retorno de valor es de cero.

Si el proceso de llamada no ha leido el acceso a la memoria especificada, el retorno del valor es un no cero. Para tener mayor información sobre el error, llame a **GetLastError**.

Si la aplicación es completada como una version anti virus, y el proceso no ha leido el acceso a la localización de la memoria especificada, la función causa una declaración e interrumpe el proceso de anti virus. Dejando el proceso de antivirus, la función continua como usual, y regresa un valor de no cero. Este comportamiento es por diseño, como una ayuda del anti virus.

Observaciones

En un ambiente frustrado de multitareas, es possible para algunos otros cabos cambiar el acceso de proceso a la memoria que está siendo probada. Aún si la función indica que el proceso ha leido el acceso a la memoria especificada, debes usar el manejo estructurado de excepción cuando intentes el acceso a la memoria. Usa el manejo estructurado de excepción que posibilita el sistema para notificar el proceso si una excepción del acceso de violación ocurre, dando al proceso una oportunidad de maejar la excepción.

Requisitos

Cliente: Requiere Windows XP, Windows 2000 Professional, Windows NT Workstation, Windows Me, Windows 98, o Windows 95.
Servidor: Requiere Windows Server 2003, Windows 2000 Server, o Windows NT Server.
Cabecera: Declarado en Winbase.h; incluye Windows.h.
Biblioteca: Use Kernel32.lib.

3.2.5 RESUMEN DE LAS FUNCIONES DE LIBRERIAS UTILIZADAS EN EL SISTEMA DE MULTIDIFUSION DE MENSAJES DE VOZ

3.2.5.1 LIBRERÍA KERNEL32.LIB

Dentro de ésta librería se utilizaron las siguientes funciones:

- GlobalAlloc
- GlobalFree
- GlobalLock
- GlobalUnlock
- CopyPTRtoBytes
- CopyBytestoPTR

GlobalAlloc. Esta función reserva el número especificado de bytes desde la pila de memoria. El manejo de memoria de Windows no provė pilas locales y globales por separado.

GlobalFree. Esta función libera la memoria reservada por GlobalAlloc y la invalida para que no pueda ser ya manipulada.

GlobalLock. Esta función asegura o bloquea la cantidad de memoria reservada por el objeto y retorna un apuntador al primer byte al bloque de memoria del objeto.

GlobalUnlock. Esta función decrementa el contador de aseguramiento asociado con la memoria del objeto que fue reservada con GMEM_MOVEABLE. Esta función no tiene efecto con la memoria de objetos reservada con GMEM_FIXED.

CopyPTRtoBytes. Esta función retorna el número de bytes de memoria reservadas apuntadas por un apuntador.

CopyBytestoPTR. Esta función hace el proceso a la función CopyPTRtoBytres, es decir devuelve la dirección de memoria especificada en bytes

3.2.5.2 LIBRERÍA MSACM32.LIB

Dentro de ésta librería se utilizaron las siguientes funciones:

- acmStreamOpen
- acmStreamClose
- acmStreamPrepareHeader
- acmStreamUnPrepareHeade
- acmStreamConvert

acmStreamOpen. Esta función abre un flujo de conversión ACM (Audio Compression Manager, Administrador de la compresión de Audio). Estos flujos de conversión son utilizados para convertir información de un formato especificado de audio a otro.

acmStreamClose. Esta función cierra un flujo de conversión ACM (Audio Compression Manager, Administrador de la compresión de Audio). Si la función es exitosa, su manejador es inhabilitado.

acmStreamPrepareHeader. Esta función prepara una estructura de la forma ACMSTREAMHEADER para un flujo de conversión ACM. Esta función debería ser llamada por cada cabecera de flujo, antes de que pueda ser usada en una conversión del flujo de audio. Una aplicación necesita preparar la cabecera del flujo de audio solo una vez para la vida del flujo. La cabecera del flujo puede ser reusada tantas veces mientras el tamaño de los buffers de fuente y destino no excedan los tamaños usados en la cabecera del flujo que originalmente se preparó.

acmStreamUnPrepareHeader. Esta función limpia la preparación realizada por la función acmStreamPrepareHeader para un flujo ACM. Esta función debe ser llamada después de que el ACM ha finalizado las operaciones de manejo de buffers. Una aplicación debería llamar a ésta función antes de liberar los buffers de origen y destino.

acmStreamConvert. Esta función solicita la conversión de un ACM a otro formato. La conversión puede ser asincrónica o sincrónica, dependiendo de cómo se haya abierto el flujo

3.2.5.3 LIBRERÍA WINMM.LIB

Dentro de ésta librería se utilizaron las siguientes funciones:

- TimeGetTime
- SndPlaySound
- WaveInOpen
- WaveInPrepareHeader
- WaveInUnPrepareHeader
- WaveInAddBuffer
- WaveInStart
- WaveInStop
- WaveInReset
- WaveInClose
- WaveOutOpen
- WaveOutPrepareHeader
- WaveOutUnOPrepareHeader
- WaveOutWrite
- WaveOutClose
- WaveOutReset

waveInAddBuffer

Esta función envía un buffer de entrada para un dispositivo de audio tipo wav. Cuando el buffer es lleno, la aplicación es notificada.

waveInStart

Esta función inicializa la entrada de un dispositivo de audio tipo wav.

waveInStop

Esta función detiene la entrada de un dispositivo de audio tipo wav.

waveInReset

La función detiene la entrada de un determinado dispositivo y reinicia la posición en curso a cero. Todos los buffers en curso son marcados como terminados y retornados a la aplicación.

waveInClose

Esta función cierra la entrada de un dispositivo de audio tipo wav.

waveOutOpen

Esta función abre el dispositivo para la grabación de un flujo de audio de salida tipo wav.

waveOutPrepareHeader

Esta función prepara un bloque de datos de tipo audio.

waveOutUnprepareHeader

Esta función limpia lo preparado por la función waveOutPrepareHeader

waveOutWrite

Esta función envía información de un bloque de datos de un especificado flujo de audio para un dispositivo de salida.

waveOutClose

Esta función cierra la salida del dispositivo para un tipo de audio

waveOutReset

Esta función reinicia el estado del dispositivo de salida para audio

waveOutPause

Esta función pausa la ejecución de un flujo de audio de salida. La posición en curso es grabada. Se usa la función waveOutRestart para reanudar la ejecución desde la posición actual.

waveOutRestart

Esta función reanuda la ejecución de un flujo de audio pausado con la función waveOutPause.

3.2.6 DESARROLLO DEL SERVIDOR DE APLICACIONES CON VB 6.0 (Visual Basic 6.0)

Anteriormente se realizó una descripción general de las funciones encapsuladas dentro de la API ***Winmm.dll*** que Windows utiliza para funciones de multimedia, y las APIs ***Kernel32.dll*** y ***Msacm32.dll*** para administración de memoria. En ésta sección se detallará desde la perspectiva de código utilizado en Visual Basic para la creación del DLL ***WaveStream.dll*** que agrupa las funciones de los APIs mencionados, y desempeñándose como un Servidor de Aplicaciones DLLs del Sistema de Multidifusión de Voz.

A continuación se muestra un extracto de la declaración de las constantes, funciones y rutinas principales dentro de ***WaveStream.dll***

```
' Declaración de constantes inicializadas con una valor Hexadecimal para representar
' los formatos de audio que soporta la aplicación

Public Const WAVE_FORMAT_PCM = &H1                              'Microsoft Windows PCM
Wave Format
Public Const WAVE_FORMAT_ADPCM = &H11                           ' ADPCM Wave Format
Public Const WAVE_FORMAT_IMA_ADPCM = &H11                        ' IMA ADPCM Wave
Format
Public Const WAVE_FORMAT_DVI_ADPCM = &H11                        ' DVI ADPCM Wave
Format
Public Const WAVE_FORMAT_DSPGROUP_TRUESPEECH = &H22   ' DSP Group Wave
Format
Public Const WAVE_FORMAT_GSM610 = &H31                          ' GSM610 Wave Format
Public Const WAVE_FORMAT_MSN_AUDIO = &H32                        ' MSN Audio Wave
Format

' Declaración de las Funciones utilizadas en multimedia
' Estas funciones son importadas desde la librería winmm.dll (API de Windows)
```

Declare Function timeGetTime Lib "winmm.dll" () As Long

Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal SoundData As Any, ByVal uFlags As Long) As Long

Declare Function waveInOpen Lib "winmm.dll" (lphWaveIn As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMATEX, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long

Declare Function waveInPrepareHeader Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInUnprepareHeader Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInAddBuffer Lib "winmm.dll" (ByVal hWaveIn As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

Declare Function waveInStart Lib "winmm.dll" (ByVal hWaveIn As Long) As Long

Declare Function waveInStop Lib "winmm.dll" (ByVal hWaveIn As Long) As Long

Declare Function waveInReset Lib "winmm.dll" (ByVal hWaveIn As Long) As Long

Declare Function waveInClose Lib "winmm.dll" (ByVal hWaveIn As Long) As Long

Declare Function waveOutOpen Lib "winmm.dll" (lphWaveOut As Long, ByVal uDeviceID As Long, lpFormat As WAVEFORMATEX, ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long

Declare Function waveOutPrepareHeader Lib "winmm.dll" (ByVal hWaveOut As Long, wH As WAVEHDR, ByVal uSize As Long) As Long

```
' Inicializa el formato de audio que se va a utilizar en la trasmisión


Private Sub InitWaveFormat(waveFmt As WAVEFORMATEX, fmtType As Long, Time Slice
As Single)
   Dim i As Long


   Select Case fmtType
   Case WAVE FORMAT ADPCM
      waveFmt.wFormatTag   WAVE FORMAT ADPCM        ' tipo de formato de audio
      waveFmt.nChannels   1                          ' número de canales - mono
      waveFmt.wBitsPerSample   4                     ' bits sample de TRUESPEECH
      waveFmt.nSamplesPerSec   c8 0kHz               ' tasa en kHz
      waveFmt.nAvgBytesPerSec   4055                 ' Bytes Seg
      waveFmt.nBlockAlign   256                      ' tamaño del bloque de información
      waveFmt.cbSize   2                                       ' extra bytes usados para
WaveFormatEx
      waveFmt.xBytes(0)   &HF9                       ' Fact Chunk - Byte 0
      waveFmt.xBytes(1)   &H1                        ' Fact Chunk - Byte 1
   Case WAVE_FORMAT_MSN AUDIO                                      ' Formato -
WAVE FORMAT MSN AUDIO
      waveFmt.wFormatTag   WAVE FORMAT MSN AUDIO     ' wave format type
      waveFmt.nChannels   1                  ' number of channels - mono
      waveFmt.wBitsPerSample   0               ' bits sample of TRUESPEECH - not used.
      waveFmt.cbSize   4                   ' extra bytes used for WaveFormatEx
      waveFmt.xBytes(0)   &H40                 ' Fact Chunk - Byte 0
      waveFmt.xBytes(1)   &H1                  ' Fact Chunk - Byte 1
   Case  WAVE FORMAT GSM610                          ' Initialize Wave Format -
WAVE FORMAT GSM610
      waveFmt.wFormatTag   WAVE FORMAT GSM610      ' wave format type
      waveFmt.nChannels   1                  ' number of channels - mono
      waveFmt.nSamplesPerSec   c8 0kHz              ' sample rate kHz
      waveFmt.nAvgBytesPerSec   1625           ' Bytes Sec
      waveFmt.nBlockAlign   65                 ' block size of data
```

```
    waveFmt.wBitsPerSample = 0               ' bits/sample of TRUESPEECH - not used.
    waveFmt.cbSize = 2                       ' extra bytes used for WaveFormatEx
    waveFmt.xBytes(0) = &H40                 ' Fact Chunk - Byte 0
    waveFmt.xBytes(1) = &H1                  ' Fact Chunk - Byte 1
  Case WAVE_FORMAT_PCM              ' Initialize Wave Format - WAVE_FORMAT_PCM
    waveFmt.wFormatTag = WAVE_FORMAT_PCM             ' format type
    waveFmt.nChannels = WAVE_FORMAT_1M08             ' number of channels (i.e. mono,
stereo, etc.)
    waveFmt.nSamplesPerSec = c8_0kHz                 ' sample rate 8.0 kHz
    waveFmt.nAvgBytesPerSec = waveFmt.nSamplesPerSec  ' for buffer estimation
    waveFmt.wBitsPerSample = 8                       ' [8, 16, or 0]
    waveFmt.nBlockAlign = waveFmt.nChannels * waveFmt.wBitsPerSample / 8 ' block size
of data
    waveFmt.cbSize = 0                               ' Not Used If [wFormatTag =
WAVE_FORMAT_PCM]
  End Select
End Sub
```

Una vez preparada la información, la aplicación procede a enviar, para el efecto utiliza la función *RecordWave*, que de detalla a continuación.

```
' Esta función graba los registros de audio en buffers de tipo String
' y envía éste buffer por medio de socket TCP/IP

Public Function RecordWave(hWND As Long, ByVal TCPSocket As Variant) As Boolean
  Dim rc As Long                             ' Function Return Code
  Dim hAS As Long                            ' ACM stream device
  Dim cWavefmt As WAVEFORMATEX               ' Wave compression format
  Dim acmHdr As ACMSTREAMHEADER              ' ACM stream header
  Dim acmHdr_x As ACMSTREAMHEADER            ' -- Double Buffering -- ACM
stream header
```

```
Code
    If Not AudioErrorHandler(rc, "waveInPrepareHeader") Then GoTo ErrorRecordWave

    ' -- Double Buffering -- Wait For Wave (xtra)Header CallBack
    Call WaitForCallBack(WaveInHDR_x.dwFlags, WHDR_PREPARED)

    ' Wait For Wave Header CallBack
    Call WaitForCallBack(WaveInHDR.dwFlags, WHDR_PREPARED)

    ' -- Double Buffering -- Add Input Wave (xtra)Buffer To Wave Input Device
    rc = waveInAddBuffer(hWaveIn, WaveInHDR_x, Len(WaveInHDR_x)) ' Validate Return
Code
    If Not AudioErrorHandler(rc, "waveInAddBuffer_x") Then GoTo ErrorRecordWave

    ' Add Input Wave Buffer To Wave Input Device
    rc = waveInAddBuffer(hWaveIn, WaveInHDR, Len(WaveInHDR)) ' Validate Return Code
    If Not AudioErrorHandler(rc, "waveInAddBuffer") Then GoTo ErrorRecordWave

    ' -- Double Buffering -- Wait For Wave (xtra)Header CallBack
    Call WaitForCallBack(WaveInHDR_x.dwFlags, WHDR_PREPARED)

    ' Wait For Wave Header CallBack
    Call WaitForCallBack(WaveInHDR.dwFlags, WHDR_PREPARED)

    Call InitWaveFormat(cWavefmt, waveCodec, TIMESLICE)   ' Set current wave format

    ' Open Configure an acm Stream Handle For Compression
    rc = acmStreamOpen(hAS, 0&, waveFmt, cWavefmt, 0&, 0&, 0&,
ACM_STREAMOPENF_NONREALTIME)
    Call AudioErrorHandler(rc, "acmStreamOpen")

    ' Initialize Audio Compression Manager Streaming Headers
    Call InitAcmHDR(hAS, acmHdr, WaveInHDR)
```

```
    Call InitAcmHDR(hAS, acmHdr x, WaveInHDR x)

    ' Prepare acm Stream Header
    rc   acmStreamPrepareHeader(hAS, acmHdr, 0&)
    Call AudioErrorHandler(rc, "acmStreamPrepareHeader")

    ' Prepare acm Stream Header
    rc   acmStreamPrepareHeader(hAS, acmHdr x, 0&)
    Call AudioErrorHandler(rc, "acmStreamPrepareHeader x")

    ' -- Double Buffering -- Wait For Wave (xtra)Header CallBack
    Call                                         WaitForACMCallBack(acmHdr x.dwStatus,
ACMSTREAMHEADER STATUSF PREPARED)

    ' Wait For Wave Header CallBack
    Call                                         WaitForACMCallBack(acmHdr.dwStatus,
ACMSTREAMHEADER STATUSF PREPARED)

    ' Start Input Wave Device Recording...
    rc   waveInStart(hWaveIn)                    ' Validate Return Code
    If Not AudioErrorHandler(rc, "waveInStart") Then GoTo ErrorRecordWave

    Do
      '   Double Buffering -- Wait For Wave (xtra)Header CallBack
      Call WaitForCallBack(WaveInHDR x.dwFlags, WHDR DONE)

      ' -- Double Buffering -- Compress acm Stream Wave Buffer
      rc   acmStreamConvert(hAS, acmHdr x, ACM STREAMCONVERTF BLOCKALIGN)
      If Not AudioErrorHandler(rc, "acmStreamConvert x") Then GoTo ErrorRecordWave

      rc   SendSoundAll(TCPSocket, acmHdr x)          ' -- Double Buffering -- Send Sound
Buffer To TCPSocket
      If Not Recording Then Exit Do                  ' Evaluate Recording Stop Flag
```

```
    ' <<Double Buffering>> Add Input Wave (xtra)Buffer To Wave Input Device
    rc = waveInAddBuffer(hWaveIn, WaveInHDR_x, Len(WaveInHDR_x)) ' Validate Return
Code
    If Not AudioErrorHandler(rc, "waveInAddBuffer_x") Then GoTo ErrorRecordWave

    Call WaitForCallBack(WaveInHDR.dwFlags, WHDR_DONE)  ' Wait For Wave Header
CallBack

    ' Convert/Compress acm Stream Wave Buffer
    rc = acmStreamConvert(hAS, acmHdr, ACM_STREAMCONVERTF_BLOCKALIGN)
    If Not AudioErrorHandler(rc, "acmStreamConvert") Then GoTo ErrorRecordWave

    rc = SendSoundAll(TCPSocket, acmHdr)          ' Send Sound Buffer To TCPSocket
    If Not Recording Then Exit Do                 ' Evaluate Recording Stop Flag

    ' Add Input Wave Buffer To Wave Input Device
    rc = waveInAddBuffer(hWaveIn, WaveInHDR, Len(WaveInHDR)) ' Validate Return Code
    If Not AudioErrorHandler(rc, "waveInAddBuffer") Then GoTo ErrorRecordWave
  Loop While Recording                            ' Continue Recording...

  ' <<Double Buffering>> UnPrepare acm Stream Header
  rc = acmStreamUnprepareHeader(hAS, acmHdr_x, 0&)
  Call AudioErrorHandler(rc, "acmStreamUnprepareHeader_x")

  ' UnPrepare acm Stream Header
  rc = acmStreamUnprepareHeader(hAS, acmHdr, 0&)
  Call AudioErrorHandler(rc, "acmStreamUnprepareHeader")

  ' Free globally allocated and locked memory variables...
  Call FreeAcmHdr(acmHdr_x)                       ' Free extra wave header memory
  Call FreeAcmHdr(acmHdr)                         ' Free wave header memory
```

```
' Close acm Stream Handle
rc   acmStreamClose(hAS, 0&)
Call AudioErrorHandler(rc, "acmStreamClose")

' << Double Buffering >> Wait For Wave (xtra)Header CallBack
Call WaitForCallBack(WaveInHDR_x.dwFlags, WHDR_DONE)

' Wait For Wave Header CallBack
Call WaitForCallBack(WaveInHDR.dwFlags, WHDR_DONE)

' Stop Input Wave Device
rc   waveInStop(hWaveIn)                    ' Validate Return Code
If Not AudioErrorHandler(rc, "waveInStop") Then GoTo ErrorRecordWave

' UnPrepare Input Wave Device Header
rc   waveInUnprepareHeader(hWaveIn, WaveInHDR, Len(WaveInHDR)) ' Validate Return
Code
If Not AudioErrorHandler(rc, "waveInUnPrepareHeader") Then GoTo ErrorRecordWave

' << Double Buffering >> UnPrepare Input Wave Device (xtra)Header
rc = waveInUnprepareHeader(hWaveIn, WaveInHDR_x, Len(WaveInHDR_x)) ' Validate
Return Code
If Not AudioErrorHandler(rc, "waveInUnPrepareHeader_x") Then GoTo ErrorRecordWave

' Close Input Wave Device
rc   waveInClose(hWaveIn)                   ' Validate Return Code
If Not AudioErrorHandler(rc, "waveInClose") Then Exit Function

' Clean Up Memory Data...
rc   FreeWaveHDR(WaveInHDR)                 ' Free Wave Header Data
rc   FreeWaveHDR(WaveInHDR_x)               ' Free Extra Wave Header Data

RecordWave = True                           ' Return Success
```

```
   RecDeviceFree   True                    ' Free Recording Device
   Exit Function                    ' Exit
'--------------------------------------------------------------
ErrorRecordWave:                              ' Clean Up Environment(Brute force no error
handling)...
'--------------------------------------------------------------
   rc   acmStreamUnprepareHeader(hAS, acmHdr, 0&)      ' Attempt To UnPrepare acm
Stream Header
   rc   acmStreamUnprepareHeader(hAS, acmHdr x, 0&)    ' Attempt To UnPrepare acm
Stream (xtra)Header
   Call FreeAcmHdr(acmHdr)                   ' Free wave header memory
   Call FreeAcmHdr(acmHdr x)                 ' Free extra wave header memory
   rc   acmStreamClose(hAS, 0&)              ' Attempt To Close acm Stream Handle

   rc   waveInStop(hWaveIn)                  ' Attempt To Stop WaveInput Device
   rc   waveInReset(hWaveIn)                 ' Attempt To Reset WaveInput Device
   rc   waveInUnprepareHeader(hWaveIn, WaveInHDR, Len(WaveInHDR)) ' Attempt To
Unprepare WaveInput Header
   rc   waveInUnprepareHeader(hWaveIn, WaveInHDR x, Len(WaveInHDR x)) ' Attempt To
Unprepare WaveInput (xtra)Header
   rc   waveInClose(hWaveIn)                 ' Attempt To Close Wave Input Device
   rc   FreeWaveHDR(WaveInHDR)                  ' Free Wave Header Data
   rc   FreeWaveHDR(WaveInHDR x)                 ' Free Extra Wave Header Data

   RecDeviceFree   True                    ' Free Recording Device
   Exit Function                    ' Exit
'--------------------------------------------------------------
End Function
```

Previo a la recepción de la llamada se utiliza la función *PlayWave* que se detalla a continuación:

```
' Función utilizada en la recepción de la llamada, es decir recepción de la información de audio

Public Function PlayWave(hWND As Long, StreamIdx As Integer) As Boolean
' La información que llega de los buffers es emitida como audio
   Dim rc As Long                          ' Function Return Code
   Dim hAS As Long                         ' ACM stream device
   Dim acmHdr As ACMSTREAMHEADER           ' ACM stream header
   Dim acmHdr_x As ACMSTREAMHEADER         ' -- Double Buffering -- ACM stream header
   Dim cWavefmt As WAVEFORMATEX            ' Wave compression format
   Dim waveFmt As WAVEFORMATEX             ' Wave format type
   Dim hWaveOut As Long                    ' Handle To A Wave Output Device
   Dim WaveOutHdr As WAVEHDR               ' Handle To A Wave Output Device Header
   Dim WaveOutHdr_x As WAVEHDR             ' Handle To A Wave Output Device Header

   Call InitWaveFormat(waveFmt, waveCodec, TIMESLICE)  ' Set current wave format

   ' Open Output Wave Device
   rc = waveOutOpen(hWaveOut, WAVE_MAPPER, waveFmt, 0&, 0&, CALLBACK_NULL)
   If Not AudioErrorHandler(rc, "waveOutOpen") Then Exit Function ' Validate Return Code

   PlayDeviceFree = False                  ' Allocate Recording Device

   ' Init Extra Wave Header Format Information
   Call InitWaveHDR(WaveOutHdr_x, waveFmt, (waveFmt.nAvgBytesPerSec * TIMESLICE))

   ' Init Wave Header Format Information
   Call InitWaveHDR(WaveOutHdr, waveFmt, (waveFmt.nAvgBytesPerSec * TIMESLICE))

   ' Prepare Output Wave Device Header
```

```
rc = waveOutPrepareHeader(hWaveOut, WaveOutHdr_x, Len(WaveOutHdr_x)) ' Validate Return Code
If Not AudioErrorHandler(rc, "waveOutPrepareHeader") Then GoTo ErrorPlayWave

' Prepare Output Wave Device Header
rc = waveOutPrepareHeader(hWaveOut, WaveOutHdr, Len(WaveOutHdr)) ' Validate Return Code
If Not AudioErrorHandler(rc, "waveOutPrepareHeader") Then GoTo ErrorPlayWave

' -- Double Buffer -- Copy (extra)Wave Data To Buffer
If Not (LoadPlayBuffer(hWaveOut, WaveOutHdr_x, waveFmt,
            PlayWaveBuffer.Stream(StreamIdx).Waves(CurPlayPos(StreamIdx)).Data,
            CurPlayPos(StreamIdx))) Then GoTo ErrorPlayWave ' Cleanup And Leave

' -- Double Buffering -- Wait For Wave (xtra)Header CallBack
Call WaitForCallBack(WaveOutHdr_x.dwFlags, WHDR_PREPARED)

' Wait For Wave Header CallBack
Call WaitForCallBack(WaveOutHdr.dwFlags, WHDR_PREPARED)

'Call InitWaveFormat(cWavefmt, waveCodec, TIMESLICE) ' Set current wave format
Call InitWaveFormat(cWavefmt, WAVE_FORMAT_PCM, TIMESLICE) ' Set current wave format

' Open Configure an acm Stream Handle For Compression
rc = acmStreamOpen(hAS, 0&, waveFmt, cWavefmt, 0&, 0&, 0&, ACM_STREAMOPENF_NONREALTIME)
Call AudioErrorHandler(rc, "acmStreamOpen")

' Initialize Audio Compression wave streaming headers...
Call InitAcmHDR(hAS, acmHdr, WaveOutHdr)
Call InitAcmHDR(hAS, acmHdr_x, WaveOutHdr_x)
```

```
' Prepare acm Stream Header
rc = acmStreamPrepareHeader(hAS, acmHdr, 0&)
Call AudioErrorHandler(rc, "acmStreamPrepareHeader")

' Prepare acm Stream Header
rc = acmStreamPrepareHeader(hAS, acmHdr_x, 0&)
Call AudioErrorHandler(rc, "acmStreamPrepareHeader_x")

' <-- Double Buffering --> Wait For Wave (xtra)Header CallBack
Call WaitForACMCallBack(acmHdr_x.dwStatus,
ACMSTREAMHEADER_STATUSF_PREPARED)

' Wait For Wave Header CallBack
Call WaitForACMCallBack(acmHdr.dwStatus,
ACMSTREAMHEADER_STATUSF_PREPARED)

' <-- Double Buffer --> Write (extra)Wave Buffer To Output Device...
rc = waveOutWrite(hWaveOut, WaveOutHdr_x, Len(WaveOutHdr_x))
If Not AudioErrorHandler(rc, "waveOutWrite_x") Then GoTo ErrorPlayWave ' Validate
Return Code

Do
   ' Copy Wave Data To Buffer
   If Not (LoadPlayBuffer(hWaveOut, WaveOutHdr, waveFmt,
                  PlayWaveBuffer.Stream(StreamIdx).Waves(CurPlayPos(StreamIdx)).Data,

                  CurPlayPos(StreamIdx))) Then GoTo CleanUpPlayWave ' Cleanup And
Leave

   ' <-- Double Buffering --> Compress acm Stream Wave Buffer
   rc = acmStreamConvert(hAS, acmHdr, ACM_STREAMCONVERTF_BLOCKALIGN)
   If Not AudioErrorHandler(rc, "acmStreamConvert") Then GoTo ErrorPlayWave
```

```
    ' Write Wave Buffer To Output Device...
    rc = waveOutWrite(hWaveOut, WaveOutHdr, Len(WaveOutHdr))
    If Not AudioErrorHandler(rc, "waveOutWrite") Then GoTo ErrorPlayWave ' Validate
Return Code

    ' -- Double Buffer -- Wait For Wave Header CallBack
    Call WaitForCallBack(WaveOutHdr_x.dwFlags, WHDR_DONE)

    ' -- Double Buffer -- Copy (extra)Wave Data To Buffer
    If Not (LoadPlayBuffer(hWaveOut, WaveOutHdr_x, waveFmt,
            PlayWaveBuffer.Stream(StreamIdx).Waves(CurPlayPos(StreamIdx)).Data,

            CurPlayPos(StreamIdx))) Then GoTo CleanUpPlayWave ' Cleanup And
Leave

    ' -- Double Buffering -- Compress acm Stream Wave Buffer
    rc = acmStreamConvert(hAS, acmHdr_x, ACM_STREAMCONVERTF_BLOCKALIGN)
    If Not AudioErrorHandler(rc, "acmStreamConvert_x") Then GoTo ErrorPlayWave

    ' -- Double Buffer -- Write (extra)Wave Buffer To Output Device...
    rc = waveOutWrite(hWaveOut, WaveOutHdr_x, Len(WaveOutHdr_x))
    If Not AudioErrorHandler(rc, "waveOutWrite_x") Then GoTo ErrorPlayWave ' Validate
Return Code

    ' Wait For Wave Header CallBack
    Call WaitForCallBack(WaveOutHdr.dwFlags, WHDR_DONE)
  Loop While Playing                    ' Continue Playing...

'---------------------------------------------------------------
CleanUpPlayWave:                        ' Cleanup...
'---------------------------------------------------------------
  ' -- Double Buffering -- UnPrepare acm Stream Header
  rc = acmStreamUnprepareHeader(hAS, acmHdr_x, 0&)
```

```
Call AudioErrorHandler(rc, "acmStreamUnprepareHeader x")

' UnPrepare acm Stream Header
rc   acmStreamUnprepareHeader(hAS, acmHdr, 0&)
Call AudioErrorHandler(rc, "acmStreamUnprepareHeader")

Call FreeAcmHdr(acmHdr)                  ' Free wave header memory
Call FreeAcmHdr(acmHdr x)                ' Free extra wave header memory

' Close acm Stream Handle
rc   acmStreamClose(hAS, 0&)
Call AudioErrorHandler(rc, "acmStreamClose")

' Wait For Wave Header CallBack
Call WaitForCallBack(WaveOutHdr.dwFlags, WHDR DONE)

' Unprepare Wave Output Buffer
rc   waveOutUnprepareHeader(hWaveOut, WaveOutHdr, Len(WaveOutHdr))

' -- Double Buffer -- Wait For Wave Header CallBack
Call WaitForCallBack(WaveOutHdr x.dwFlags, WHDR DONE)

' -- Double Buffer -- Unprepare Wave Output Buffer
rc   waveOutUnprepareHeader(hWaveOut, WaveOutHdr x, Len(WaveOutHdr x))

' Close Output Wave Device
rc   waveOutClose(hWaveOut)
If Not AudioErrorHandler(rc, "waveOutClose") Then Exit Function ' Validate Return Code

' Clean Up Memory Data...
rc = FreeWaveHDR(WaveOutHdr)             ' Free Wave Header Data
rc   FreeWaveHDR(WaveOutHdr x)           ' Free Extra Wave Header Data
```

```
    PlayWave = True                          ' Return Success
    PlayDeviceFree = True                    ' Free Recording Device
    Exit Function                            ' Exit
'-------------------------------------------------------------
ErrorPlayWave:                               ' Handle Errors And Cleanup...
'-------------------------------------------------------------
    rc = acmStreamUnprepareHeader(hAS, acmHdr, 0&)        ' Attempt To UnPrepare acm
Stream Header
    rc = acmStreamUnprepareHeader(hAS, acmHdr_x, 0&)      ' Attempt To UnPrepare acm
Stream (xtra)Header
    Call FreeAcmHdr(acmHdr)                  ' Free wave header memory
    Call FreeAcmHdr(acmHdr_x)                ' Free extra wave header memory
    rc = acmStreamClose(hAS, 0&)             ' Attempt To Close acm Stream Handle

    rc = waveOutUnprepareHeader(hWaveOut, WaveOutHdr, Len(WaveOutHdr)) ' Attempt To
Unprepare Header
    rc = waveOutUnprepareHeader(hWaveOut, WaveOutHdr_x, Len(WaveOutHdr_x)) '
Attempt To Unprepare Header
    rc = waveOutClose(hWaveOut)              ' Close Wave Output Device
    rc = FreeWaveHDR(WaveOutHdr)             ' Free Wave Header Data
    rc = FreeWaveHDR(WaveOutHdr_x)           ' Free Extra Wave Header Data

    PlayDeviceFree = True                    ' Free Recording Device Flag
    Exit Function                            ' Exit
'-------------------------------------------------------------
End Function
```

CAPITULO IV

VALIDACIÓN DEL PROYECTO

4.1 CONCLUSIONES

Al finalizar el desarrollo de esta disertación de tesis hemos podido extraer las siguientes conclusiones:

- ✓ El software de multidifusión de mensajes de voz fue realizado en el lenguaje de programación Visual Basic 6.0; gracias a este paquete de programación se permitió que permitió utilizar todas sus bondades, tales como sus librerías (Kernel32.dll; Gdi32.dll; User32.dll) que facilitan la comunicación en red.

- ✓ Se utilizó el puerto paralelo del computador para la conexión del hardware que se fabricó para conectividad de varios parlantes. Dicho puerto nos facilitó la transmisión de voz en tiempo real, gracias a su trabajo con el sistema binario, y de esta manera se logró codificar la voz en 0 y 1 (Los datos son enviados en sistema binario y a la salida del computador nos entrega datos en decimal).

- ✓ Al realizar una aplicación cliente – servidor se permitió no solo a las computadoras que forman parte de la red de la Escuela de Sistemas de la PUCESA estar integradas por el Sistema de Multidifusión de Mensajes de Voz; sino que el Software es ampliable a toda la red de la PUCESA, gracias a su compatibilidad entre sistemas operativos (Windows 98, Windows XP); y se puede utilizar cualquier máquina como servidor.

- ✓ La multidifusión se vio limitada a la transmisión de voz por un solo canal debido a la condición del decodificador que convierte los números binarios a decimales, y por esta característica únicamente puede activar el pin seleccionado en el software, mas no realiza la sumatoria para enviar a todos los pines en forma colectiva, lo realiza en forma individual.

- ✓ La utilización del lenguaje de programación (Visual Basic 6.0) en su estructura orientado o objetos, ya facilita la creación de módulos independientes pero que se interrelacionan entre sí y evitar la redundancia de código.

- ✓ Al fabricar un hardware de integración entre la Computadora y los parlantes se utilizó conocimientos de electrónica y lógica matemática; lo cual nos sirvió para implementar conocimientos adquiridos en semestres anteriores.

4.2 RECOMENDACIONES

- ✓ Se recomienda a las personas que deseen programar alguna aplicación cliente – servidor, utilizar el lenguaje de programación Visual Basic, que gracias a tener librerías incluidas para este fin se facilita el trabajo y lo optimiza.
- ✓ Es muy práctico y útil la utilización del puerto paralelo del computador para el envió de datos. Es recomendable para las personas que requieren este servicio, ya sea voz, audio o datos; utilizar dicho puerto, que facilita el envío y entrada de información.
- ✓ Se recomienda a las generaciones venideras que realicen temas de disertación, los realicen con una funcionabilidad en red, en tal virtud que todo computador pueda acceder a las bondades del mismo. Para dicho propósito es loable que el software sea compatible con todas las versiones de sistemas operativos existentes en la entidad en donde se lo va a instalar.

- ✓ Para que la multidifusión no se vea limitada al envío de voz por un solo canal, se recomienda adquirir un decodificador de mejores características y con mayor potencia, de esta manera se logrará en envío de voz en forma individual y colectiva por los canales, también se incrementará el número de salidas o pines en el hardware construido.
- ✓ Lo más recomendable al momento de realizar una aplicación cliente – servidor es evitar una redundancia de código, es por ese motivo que es preferible programar en un lenguaje orientado a objetos de esta manera a más de obtener pantallas amigables al usuario, se optimiza el código fuente.
- ✓ Siempre hay que utilizar los conocimientos adquiridos durante la carrera y de esta manera se puede implantar en bien de la sociedad.

4.3 VALIDACION

Ambato abril 1, 2004
PRO-149-04-PUCESA

Ingeniero
Telmo Viteri
DIRECTOR ESCUELA DE SISTEMAS
Pontificia Universidad Católica de Ambato
Presente

Ingeniero Viteri:

Por la presente me permito indicar que luego de haber examinado el trabajo realizado por la señora Rina Katherine Sánchez Reinoso, alumna de la Escuela de Sistemas referente al Sistema de Multidifusión de mensajes de voz, considero que es aceptable y útil este software, por lo que se puede dar paso a la defensa de este proyecto.

Por la gentil atención, reciba mi agradecimiento.

Atentamente,

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR SEDE AMBATO PRO-RECTORADO

P. Dr. César González Loor
PRO-RECTOR PUCE SEDE AMBATO

cc.: archivo

PONTIFICIA
UNIVERSIDAD
CATOLICA
DEL ECUADOR
SEDE AMBAT
PRO-RECTO

Av. Manuelita Sáenz s/
Sector El Tropezón
Apartado Postal No.18-
Telf: 593 3 411 868 ext.
Fax: 593 3 411 868 ext.
pucesedeambato@hotm
Ambato - Ecuador
www.pucesa.edu.ec

Ambato 01 de Abril de 2004
Of.089-EAE-PUCESA-04

Ingeniero
Telmo Viteri
DIRECTOR ESCUELA DE SISTEMAS
PUCE SEDE AMBATO
Presente

Ingeniero Viteri:

La presente tiene como fin comunicarle que luego de haber examinado el trabajo realizado por la señora Rina Katherine Sánchez Reinoso, alumna de la escuela de sistema referente al sistema de multidifusión de mensajes de voz, considero que es aceptable y útil este software, por lo que se puede dar paso a la defensa de este proyecto.

Por la atención a la presente, y en espera de su respuesta favorable agradezco y suscribo.

Atentamente,

Ing. Vinicio Mejia Vayas
DIRECTOR

VMV/rr.

c.c. archivo

Ambato abril 1, 2004
PRO-150-04-PUCESA

Ingeniero
Telmo Viteri
DIRECTOR ESCUELA DE SISTEMAS
Pontificia Universidad Católica de Ambato
Presente

Ingeniero Viteri:

Por la presente me permito indicar que luego de haber examinado el trabajo realizado por el señor Francisco Xavier Medina Bravo, alumno de la Escuela de Sistemas referente al Sistema de Multidifusión de mensajes de voz, considero que es aceptable y útil este software, por lo que se puede dar paso a la defensa de este proyecto.

Por la gentil atención, reciba mi agradecimiento.

Atentamente,

PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
SEDE AMBATO
PRO-RECTORADO

P. Dr. César González Loor
PRO-RECTOR PUCE SEDE AMBATO

cc.: archivo

PONTIFICIA
UNIVERSIDAD
CATOLICA
DEL ECUADOR
SEDE AMBA
PRO-RECT

Av. Manuelita Sáenz s/
Sector El Tropezón
Apartado Postal No.18
Telf: 593 3 411 868 ex
Fax: 593 3 411 868 ext
pucesedeambato@hotm
Ambato - Ecuador
www.pucesa.edu.ec

Ambato 01 de Abril de 2004
Of.089-EAE-PUCESA-04

Ingeniero
Telmo Viteri
DIRECTOR ESCUELA DE SISTEMAS
PUCE SEDE AMBATO
Presente

Ingeniero Viteri:

La presente tiene como fin comunicarle que luego de haber examinado el trabajo realizado por el señor Francisco Xavier Medina Bravo, alumno de la escuela de sistema referente al sistema de multidifusión de mensajes de voz, considero que es aceptable y útil este software, por lo que se puede dar paso a la defensa de este proyecto.

Por la atención a la presente, y en espera de su respuesta favorable agradezco y suscribo.

Atentamente,

Ing. Vinicio Mejía Vayas
DIRECTOR

VMV/rr.

c.c. archivo

GLOSARIO DE TERMINOS

- ✓ **A / D.-** Analógico / Digital.
- ✓ **ACM.-** Audio Compression Manager, Administrador de la Comprensión de Audio.
- ✓ **ADDOSI.-** Análisis y diseño orientado o objetos de sistemas de información.
- ✓ **API.-** Aplication Program Inteface, Interface del programa de Aplicación
- ✓ **APR.-** Apache Portable Runtime, Carrera temporal portátil Apache.
- ✓ **ARPANET.-** Advenced Research Projects Agency Network, Investigación de Proyectos avanzados en agencia de Redes.
- ✓ **ASP.-** Advanced Signal Processor, Procesador de Señal Avanzado.
- ✓ **ATM.-** Asynchronous Transfer Mode, Modo de Tranferencia Asincrónica.
- ✓ **BIOS.-** Basic Input / Output System, Sistema Básico de Entrada / Salida.
- ✓ **CBR.-** Constant Bit Rate
- ✓ **CTI.-** Center Technology Including, Centro de Tecnología Incluido.
- ✓ **DB25.-** Data Binding 25, Información Requerida 25.
- ✓ **DLL.-** Dinamic Link Libraries, Acceso Dinámico a Librarías.
- ✓ **DSP.-** Digital Signal Processor, Procesador Digital de Señales.
- ✓ **I / O.-** Input / Output.
- ✓ **IP.-** Internet Protocol, Protocolo de Internet.
- ✓ **ISDN.-** Integreted Service data Network, Red Integrada de Servicio de Información.
- ✓ **ISP.-** Internet Service Provider, Proveedor de servicio de Internet.
- ✓ **ITSP.-** Internet Telephony Service Provider, Proveedor Telefónico de Servicio de Internet.
- ✓ **ITU.-** Internacional Telecommunication Union, Union Nacional de Telecomunicaciones.
- ✓ **LAN.-** Local Area network, Red de Area Local.
- ✓ **LPT1.-** Local port 1, Puerto Local 1.
- ✓ **MAN.-** Metropolitan Area Network, Red de Area Metropolitana.
- ✓ **MCU.-** Multimedia Conference Unit, Unidad de Conferencia de Multimedia.
- ✓ **MIDI.-** Musical Instrumental Digital Interface, Interface digital de música instrumental.

- ✓ **MPEG.-** Moving Picture Experts Group, Grupo experto en dibujos y películas.
- ✓ **OLE.-** Object Linking and Embedding, Incrustación y Vinculación de Objetos.
- ✓ **PBX.-** Private Branch Exchange, Sucursal Privada de Intercambio.
- ✓ **PC.-** Personal Computer, Computadora Personal.
- ✓ **PRI.-** Paving Roofing Industrial.
- ✓ **PRI.-** Paving Roofing Industrial.
- ✓ **RDSI.-** Red Digital de Servicio Integrado.
- ✓ **ROM.-** Read Only Memory, Memoria Solo de Lectura.
- ✓ **ROM.-** Real Only Memory, Memoria Solo de Lectura.
- ✓ **RSVP.-** Resourse Reservation Protocol.
- ✓ **RTB.-** Red de Telefonía Básica.
- ✓ **RTCP.-** Real Time Control Protocol, Protocolo de Control en Tiempo Real.
- ✓ **RTP.-** Real Time Transport Protocol, Protocolo de Transporte en Tiempo Real.
- ✓ **RTSP.-** Real Time Straming Protocol.
- ✓ **SIMM.-** Single in – Line Memory Module, Unico Modulo de Memoria en Linea.
- ✓ **SMDV.-** Sistema de Multidifusión de mensajes de voz.
- ✓ **TCP/IP.-** Transmisión Control Protocol / Internet Protocol, Control del Protocolo de Transmisión / Protocolo de Internet.
- ✓ **UDP.-** User Datagram Protocol, Protocolo del datagrama usado.
- ✓ **VoIP.-** Voz sobre IP.
- ✓ **VTOA.-** Voz Sobre ATM
- ✓ **WAN.-** Wide Area Network, Red de Area Extensa..

BIBLIOGRAFIA

- ✓ BERSON, A. (1996). *Client / Server Architecture*. Segunda Edición. Ed. Mc.Graw – Hill.

- ✓ FRISH, S. *"Curso de Física General"*, Cuarta Edición, Tomo 1, Página 467. Editorial MIR, Año 1981.

✓ TIMOREVA, A. *"Introducción a la Electrónica"*, Segunda Edición, Tomo I, Página 236. Editorial MIR, Año 1990.

✓ ALONSO, Marcelo. *"Física"*, 21a Edición, Tomo 2, Página 31. Editorial CULTURAL, Año 1978.

✓ PRESSMAN. *"Ingeniería de Software"*, 2da Edición, Tomo 1, Página 68. Editorial MIR, Año 1988.

✓ SENN, James. *"Análisis y Diseño de Sistemas"*, 1ra Edición, Tomo 1, Página 64. Editorial CULTURAL, Año 1978.

✓ CENTRO DE COMPUTO DE LA PUCESA, *Organización y funcionamiento del Centro de Cómputo.*

DIRECCIONES DE INTERNET

- http://www.elruido.com
- http://www.ansi.org
- http://www.monografias.com.
- http://www.lafacu.com
- http://www.iec.org/online/tutorials/
- http://elvex.ugr.es/decsai/builder/appendix/components/Internet.html
- http://delphipage.free.fr/internetc.html
- http://www.vozsobreip.com.br
- http://www.cybercursos.net/cursos-online/foxpro/vfpartic.htm
- http://www.programming-vb.com/vb/api/api.htm
- http://www.ip.com
- http://www.tcp.com

ANEXOS

ANEXO 1

ANEXO 2

Sistema de Multidifusión de Voz

MANUAL DE USUARIO

1. INTRODUCCIÓN

El Sistema de Multidifusión de Voz (SMDV) es un sistema compuesto por elementos de Hardware que son administrados por su Software. Este Sistema permite enviar mensajes de voz desde un computador que disponga multimedia hacia diferentes parlantes (hasta nueve) ubicados físicamente en diferentes sitios, pero con una conexión centralizada en un determinado computador a través del hardware del sistema de Multidifusión de Voz. Por otro lado, el software permite enviar mensajes desde cualquier computador con multimedia conectado a una red de computadoras que utilice el protocolo TCP/IP. La descripción de lo antes dicho se muestra en la Figura 1.

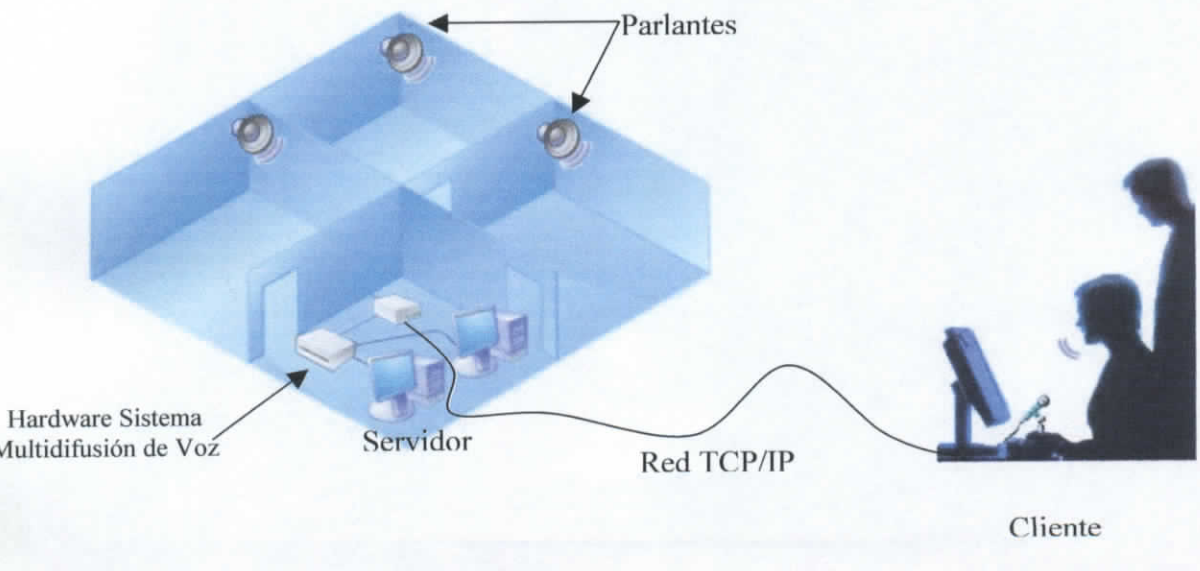
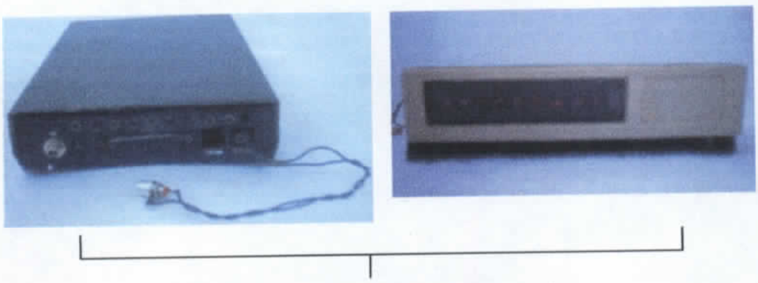


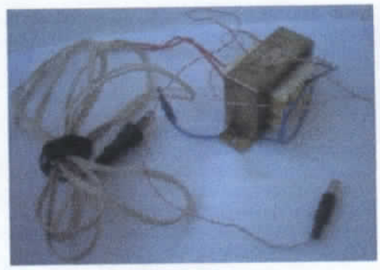
FIGURA 1. Estructura General del Sistema de Multidifusión de Voz

2. HARDWARE DEL SISTEMA DE MULTIDIFUSION DE VOZ

El Hardware del Sistema está compuesto de un dispositivo electrónico que permite canalizar el sonido emitido por un computador a través de su tarjeta multimedia, hacia uno de los nueve parlantes que permite el dispositivo. Los diferentes elementos que componen éste dispositivo se muestra en la Figura 2.

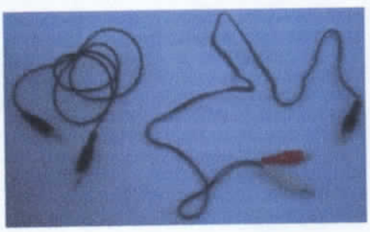


(a) Dispositivo de distribución

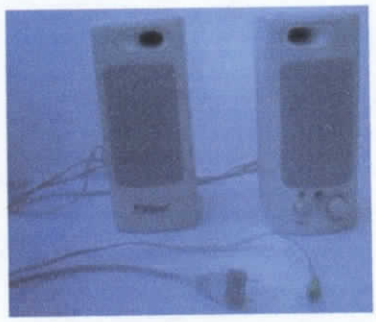


(b) Fuente de poder

(c) Cable dispositivo-computador (salida de datos)



(d) Cable dispositivo-computador (salida de sonido)



(e) Parlantes

(f) Micrófono

FIGURA 2. Componentes de Hardware del Sistema de Multidifusión de Voz

2.1 Instalación del Hardware

En el dispositivo de distribución (a) se conectan los componentes: (b) fuente de poder, (c) Cable de salida de datos del computador a través del puerto paralelo, la parte gris va en el dispositivo de distribución y la parte beige en el puerto del computador, (d) el cables de salida de sonido del computador a través de la tarjeta multimedia hacia el dispositivo de distribución, en la gráfica se muestran cualquiera de los dos cables que se pueden utilizar, (e) los parlantes se conectan en cualquiera de las nueve entradas del dispositivo de distribución, el micrófono (f) debe conectarse en el computador. En la Figura 3 se muestran las entradas del dispositivo de distribución.

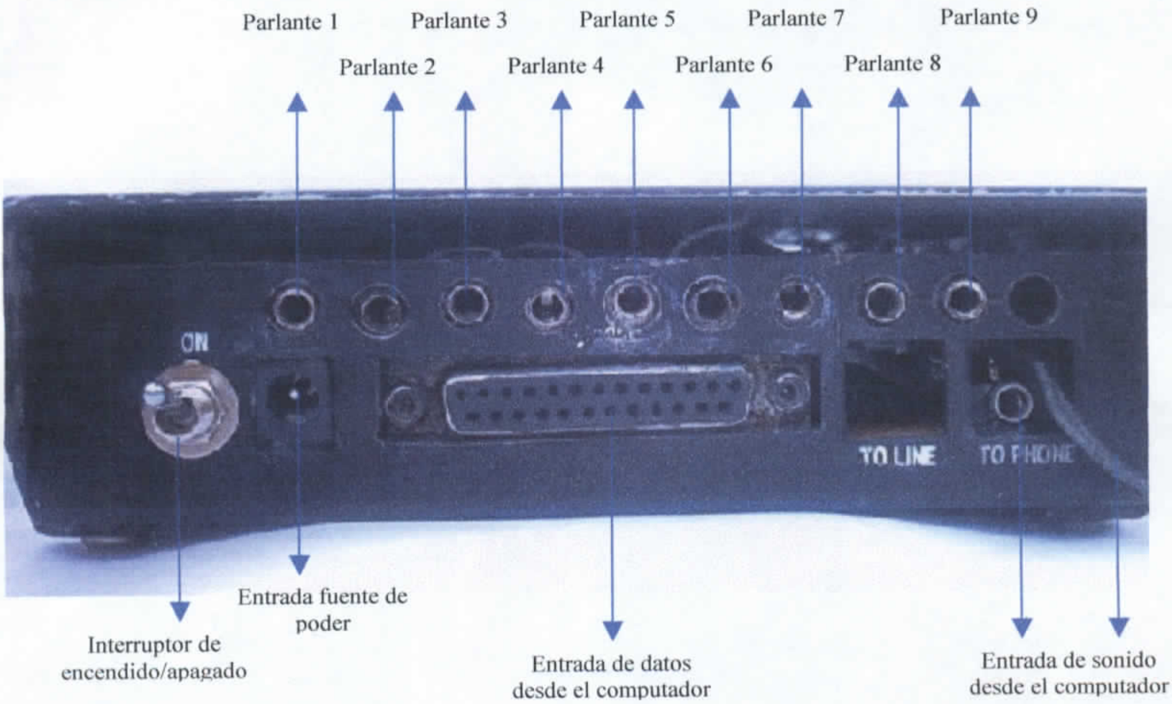


FIGURA 3. Puertos del dispositivo de distribución

En la Figura 4 se muestra la interconexión básica de todos estos elementos.

FIGURA 4. Elementos de Hardware interconectados

Es importante recalcar que el dispositivo de distribución estará conectado a una computadora que servirá de servidor, y una vez conocida su ubicación, en lo posterior se conectaran los nueve parlantes hacia las diferentes ubicaciones.

3. SOFTWARE DEL SISTEMA DE MULTIDIFUSION DE VOZ

El Software del Sistema está compuesto básicamente por dos programas ejecutables, dos librerías dinámicas y dos archivos de configuraciones, tal como se muestra en la Tabla 1:

Archivo	Tipo	Tamaño	Descripción
Servidor.exe	Ejecutable	328Kb	Programa para la administración del dispositivo de distribución.
Cliente.exe	Ejecutable	244Kb	Programa para enviar mensajes de voz desde un cliente de red
WaveStream.dll	Librería	36Kb	Librería que permite manipular APIs de multimedia de MS. Windows
inpout32.dll	Librería	32Kb	Librería que permite manipular funciones para la salida/entrada de datos a través del Puerto Paralelo (Sólo es usada por Servidor.exe)
config.ini	Configuración	1KB	Archivo de texto que almacena la configuración de los nueve parlantes (Es leído únicamente por el Servidor.exe)
configip.ini	Configuración	1KB	Archivo de texto que almacena direcciones IP de conexión, usadas en Servidor.exe y Cliente.exe

Tabla 1. Componentes de Software del Sistema de Multidifusión de Voz

3.1 Instalación del Sofware

En el CD del Sistema de Multidifusión de Voz existe la carpeta *Multidifusion*, en la que se guardan los archivos mencionados en la Tabla 1. Dentro de ésta carpeta existen los directorios *Servidor*, y *Cliente*, que almacenan el código fuente del sistema, que por hoy no nos interesa ya que se va a utilizar los programas ya compilados.

El Sistema de Multidifusión de Voz fue desarrollado y probado en la plataforma MS Windows XP Professional Edition y por lo tanto a continuación se va a explicar la instalación en dicha plataforma.

3.1.1 Instalación del Servidor

El programa Servidor.exe permite emitir mensajes de voz desde el computador que tiene conectado el hardaware de multidifusión de voz al puerto paralelo y a la tarjeta se salida de multimedia. Permite además recibir solicitudes de clientes con opciones de parlantes a emitir un mensaje de voz que se envía por una red IP, es decir voz en tiempo real. Esto último mediante un programa llamado Cliente.exe instalado en computadores remotos en red con disponibilidad de sistema multimedia.

Antes de poner a ejecutar el programa Servidor.exe en el computador que tendrá el conectado el hardaware de multidifusión de voz, es necesario realizar los sisguientes pasos:

a) Copiar en C:/Windows/System las librerías WaveStream.dll y inpout32.dll que se encuentran en Multidifusion/Servidor

En caso de un error en VOZ sobre IP (es decir error en la librería WaveStream.dll o Activex), es necesario generar nuevamente la librería WaveStream.dll en el computador que se quiere instalar. La forma de hacerlo es abrir el archivo de proyecto WAVEDLL.VBP dentro de la carpeta Multidifusion/Servidor/WAVESTRM, para esto debe estar instalado Microsoft Visual Studio 6.0.

Una vez dentro de MS Visual Estudio, seleccionar el menú ARCHIVO->Generar WaveStream.dll, y guardar la librería en c:/Windows/system

La libreria inpout32.dll (32K) está generada para Windows XP/2000/ME, pero si se va a instalar en una versión inferior de Windows, se debe utilizar la librería inpout32.dll (27K) que se encuentra en Multidifusion/Servidor/Otros/inpoutwin98

b) El programa Servidor.exe lee la configuración de parlantes desde el archivo config.ini, por tal motivo es necesario que éste archivo se encuentre en la misma carpeta donde esté el ejecutable.

c) De igual forma, el programa Servidor.exe lee la configuración de las diferentes direcciones IP que puede utilizar, por tal motivo es necesario que el archivo configip.ini, se encuentre en la misma carpeta donde se encuentra Servidor.exe.

3.1.2 Instalación del Cliente

El programa Cliente.exe es el que permite enviar la solicitud al programa Servidor.exe para emitir un mensaje de voz sobre IP enviándole el parlante al cual se quiere emitir.

El programa Cliente.exe debe ejecutarse en un equipo diferente al que se ejecuta el servidor para evitar conflictos de uso de puertos de WINSOCKET.

En la instalación del cliente se debe considerar lo siguiente:

a). Para el cliente se necesita únicamente la librería WaveStream.dll en la carpeta c:/windows/System del computador. No se necesita la librería inpout32.dll en el cliente, ya que no se envía un comando al puerto paralelo del cliente, sino al puerto del socket del servidor.

El procedimiento de copia de la librería WaveStream.dll se lo hace de la misma forma que se explicó en la instalación del servidor

b) El programa Cliente.exe, necesita únicamente el archivo configip.ini, que almacena las direcciones IP posibles de conexión. El archivo de configuraciones config.ini, no es necesario ya que la configuración es enviada desde el Servidor, unicamente cuando el cliente se conecta. Lo anterior con el fin de evitar configuraciones desactualizadas.

4. MODO DE USO DEL SISTEMA DE MULTIDIFUSION DE VOZ

El Sistema de Multidifusión de Voz puede funcionar en formas local o en forma remota, es decir que se podría emitir mensajes de voz directamente desde el computador donde se encuentra conectado el hardware del sistema de multidifusión de voz o desde un cliente remoto, ya sea una Intranet o incluso desde Internet.

A continuación se explica el modo de uso del Sistema, se comenzará con la parte del Servidor y luego la parte de Cliente.

4.1 Usando el Servidor

Previo a la ejecución del programa Servidor (Servidor.exe) es necesario la instalación del hardware descrito en la sección 2.1. Con éste requerimiento se procede a ejecutar el archivo Servidor.exe, y aparecerá una interfaz gráfica que permite escoger algunas opciones, tal como se muestra en la Figura 5.

FIGURA 5. Interfaz principal del Servidor del Sistema SMDV.

En la interfaz que se muestra al ejecutar el programa Servidor.exe existen cuatro botones: [Servidor], [Local], [Configuración] y [Salir]. A continuación se explican las actividades que se pueden realizar con cada una de ésta opciones, se comenzará desde la última opción.

4.1.1 Opción [Salir]

Al presionar el botón [Salir] se termina la ejecución del programa Servidor.exe, lo que impide que clientes puedan solicitar el servicio de emisión de mensajes.

4.1.2 Opción [Configuración]

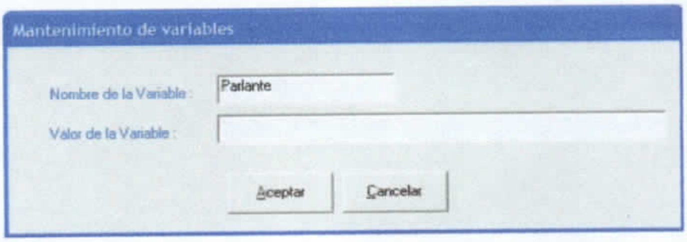


FIGURA 7. Cuadro de diálogo para crear una nueva variable

Es importante recalcar que el Sistema está programado para leer variables de la forma Parlante0#, es decir si la variable toma un nombre diferente, la apliación no la reconocerá.

Al presionar el botón 

aparecerá un cuadro de diálogo en la que se puede modificar el nombre y la descripción de una variable existente, tal como se muestra en la Figura 8.

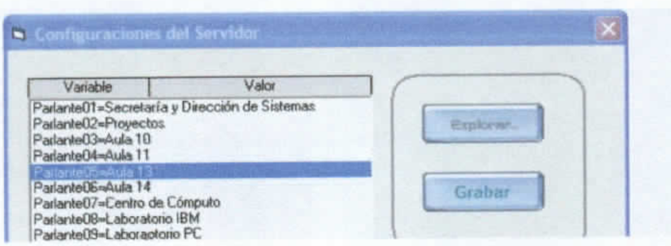
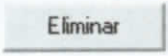


FIGURA 8. Cuadro de diálogo para modificar una variable

Básicamente, ésta opción es utilizada en el caso de que algún paralante se cambie de ubicación y se pueda indicar a la variable la nueva descripción.

Al presionar el botón 

eliminará la variable que se encuentre seleccionada en la lista.

Por otro lado, en la parte derecha de la pantalla se tienen tres botones [Explorar], [Grabar] y [Cerrar].

El botón [Explorar] permite leer la configuración de algún otro archivo diferente al config.ini y lo pone en la lista. Para este efecto se abre una ventana de diálogo en la que se permite seleccionar un archivo de texto que contenga la configuración, tal como se muestra en la Figura 9.

Se debe tomar en cuenta que ésta nueva configuración se almacenará permanentemente en el archivo config.ini al momento que se presione el botón [Grabar].

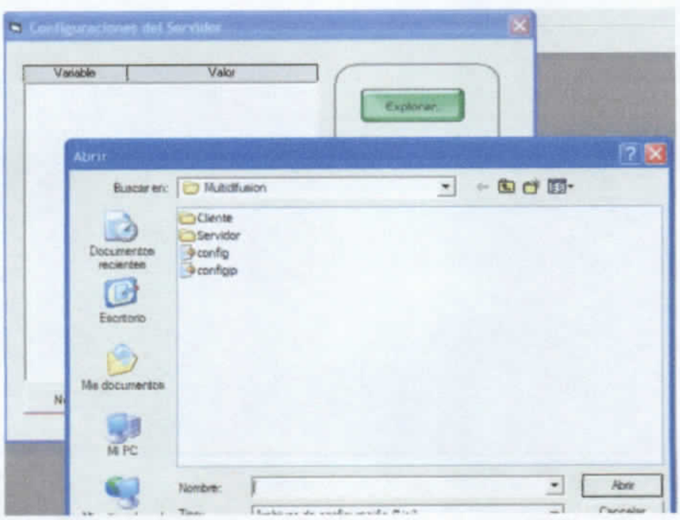


FIGURA 9. Cuadro de diálogo para buscar archivo de configuración

El botón [Grabar] permite almacenar de forma permanente en el archivo config.ini, toda aquella información que se encuentre en la lista. Si el proceso de grabación fue exitoso, entonces aparecerá un mensaje indicando ésta información.

Finalmente el botón [Cerrar] permite cerrar la ventana de configuraciones. Si se presiona el botón [Salir] sin haber presionado el botón [Grabar], no se grabarán los cambios que se hayan realizado.

4.1.3 Opción [Local]

Volviendo a la interfaz principal, al presionar el botón [Local] se abrirá una ventana en la que se puede enviar mensajes directamente desde el computador al que está conectado el hardware del sistema de multidifusión de voz (SMDV). Esta ventana se muestra en la Figura 10.

Únicamente se necesita señalar el parlante al cual se quiere emitir un mensaje y el software automáticamente enviará una señal al puerto paralelo, para que sea leído por el hardware el SMDV y canalice el sonido hacia ese parlante. En la parte inferior se puede observar el comando que se envía al puerto paralelo. Una vez seleccionado el parlante, ya se puede emitir el mensaje con el micrófono que debe estar conectado a la tarjeta de sonido del computador.

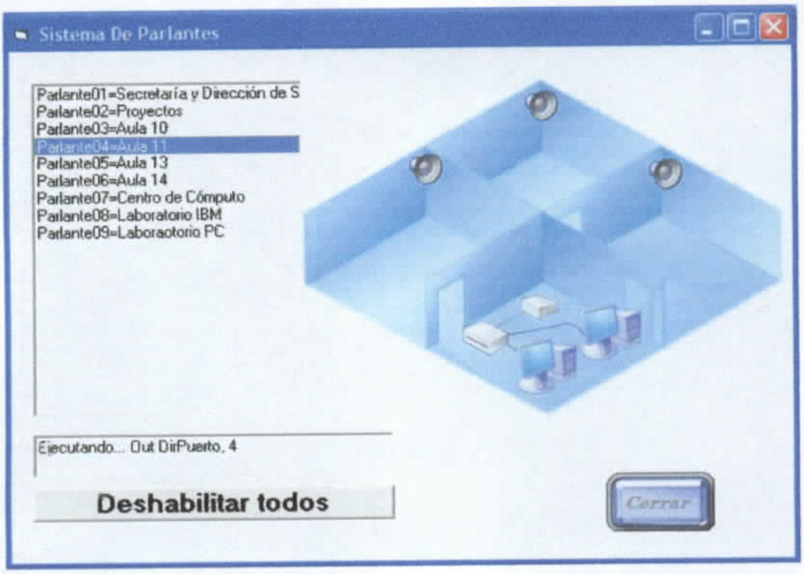


FIGURA 10. Ventana para enviar mensajes de voz en forma local

La pantalla dispone de un botón [Deshabilitar todos] que permite que se "encere" el puerto paralelo, es decir no se emitirá el sonido a ningún parlante.

Finalmente el botón [Cerrar] cerrará la ventana para emisión de mensajes en forma local.

4.1.4 Opción [Servidor]

La última opción de botón de la pantalla principal [Servidor], es una de las más importantes, ya que permite la funcionalidad de envío de mensajes en forma remota. Al presionar el botón aparecerá la pantalla que se muestra en la Figura 11.

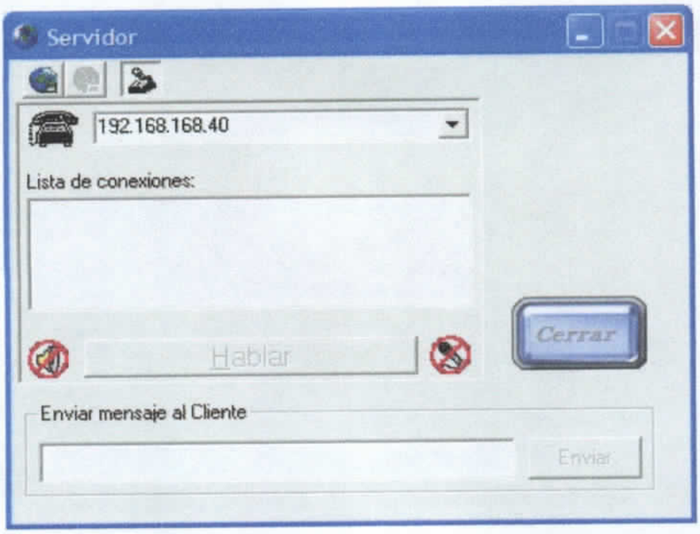


FIGURA 11. Ventana para esperar conexión es de clientes remotos

Al momento de ejecutar la pantalla que se muestra en la Figura 11 automáticamente se activa un proceso que permite escuchar solicitudes de clientes. Lo único que se necesita para que los clientes puedan enviar mensajes de voz, es que ésta pantalla se encuentre abierta. Sin embargo, se ha dejado algunas funcionalidades extras que se puede realizar a través de ésta pantalla, como se describe a continuación.

Se puede seleccionar una de las direcciones IP de la *Lista de conexiones*, y luego presionar el botón . Esto permitirá conectarse a un cliente con el que se puede comunicar mediante voz, es decir puede utilizarse como un CHAT. El requisito es que el cliente tenga abierta la pantalla similar a ésta que se describirá en la parte de modo de uso de cliente.

Este Chat, permite la conexión con varios usuarios a la vez, que se listan en *Lista de conexiones*, y se puede enviar un mensaje de voz sobre IP, seleccionando la dirección y manteniendo presionado el botón [Hablar] mientras se emite el mensaje, tal como se muestra en la Figura 12.

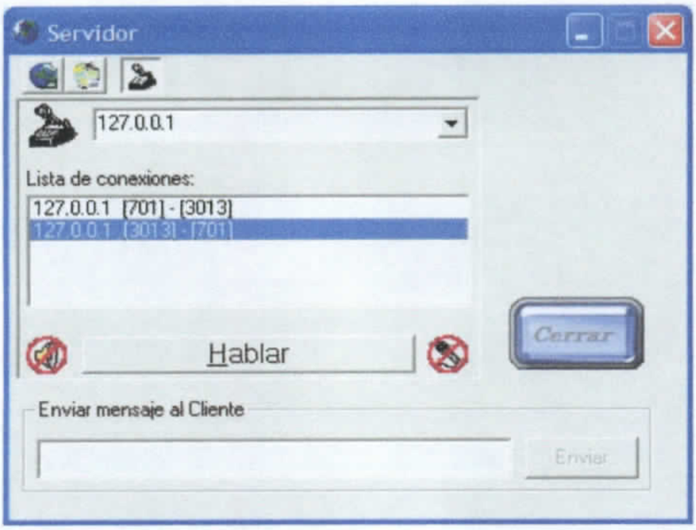


FIGURA 12. Servidor conectado a un usuario, se indica dirección IP y puerto de entrada y salida

Cuando se quiera desconectar a uno de los interlocutores, simplemente se selecciona de la *Lista de conexiones* y se presiona el botón 

Finalmente, el botón  permite seleccionar o deseleccionar la opción de contestado automático, es decir si está presionado el servidor atenderá automáticamente cualquier "llamada" de clientes, en caso contrario si no está presionado el botón emitirá una pregunta si desea recibir la llamada de un determinado cliente.

El recuadro de abajo *Enviar mensaje al cliente*, se activará únicamente cuando un cliente se conecte al servidor, y se podrá enviar un mensaje de texto al cliente. Cuando un cliente se conecte se indicará en el título de la ventana y en el mensaje *Enviar mensaje al cliente nombre_del_cliente*. Esta última opción puede ser utilizada para solicitar a un cliente que cierre su ventana de comunicación, para permitir que otros clientes se conecten, ya que para el envío de mensajes remotos se permite un solo cliente a la vez, por razones obvias de no permitir una mezcla de mensajes, pero no como cuando se utiliza el servidor como chat de voz, ya que ahí si se puede tener múltiples clientes.

Es importante recalcar que con el botón [Cerrar] sólo se esconde la ventana del servidor que escucha solicitudes, es decir se puede navegar por las ventanas de configuraciones y local mientras el servidor sigue recibiendo solicitudes. Únicamente cuando se pone la opción [Salir] de la ventana principal, el servidor ya no atenderá solicitudes.

4.2 Usando el Cliente

Un computador con disponibilidad de sistema multimedia podrá ejecutar el programa 

(Cliente.exe), con las condiciones explicadas en la instalación. Al ejecutar el programa se mostrará la ventana que se muestra en la Figura 13.

FIGURA 13. Interfaz principal del Cliente del Sistema SMDV.

En la interfaz que se muestra al ejecutar el programa Cliente.exe existen dos botones: [Cliente] y [Salir]. A continuación se explican las actividades que se pueden realizar con cada una de ésta opciones, se comenzará desde la última opción.

4.2.1 Opción [Salir]

Al presionar el botón [Salir] se termina la ejecución del programa Cliente.exe, y se cierran la conexión con el servidor, si estuviere presente.

4.2.2 Opción [Cliente]

Esta opción [Cliente], es la que permite la conexión con el programa servidor que dispone el hardware del SMDV, y de ésta manera enviar mensajes en forma remota. Al presionar el botón aparecerá la pantalla que se muestra en la Figura 14.

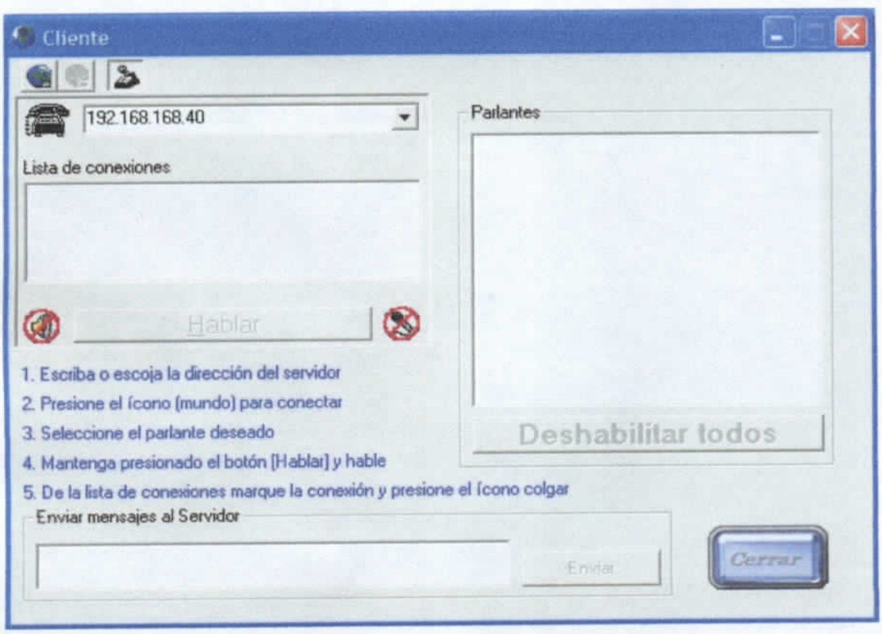


FIGURA 14. Ventana cliente para conectarse con el servidor

Esta pantalla muestra una configuración similar a la pantalla principal del servidor, a excepción de una lista de *Parlantes* que aparece vacía y deshabilitada

El funcionamiento también es muy similar, es decir se puede seleccionar una de las posibles direcciones IP del Servidor de la *Lista de conexiones*, y luego presionar el botón . Esto permitirá conectarse al servidor. Si la conexión se estableció con éxito aparecerá en el título de la ventana, el nombre del servidor a donde se conectó, y además se habilitarán las opciones que estaban deshabilitadas inicialmente, de la misma forma la lista de parlantes se llenará con la configuración del archivo config.ini, pero que se encuentra en el computador donde se ejecuta el servidor, tal como se muestra en la Figura 15.

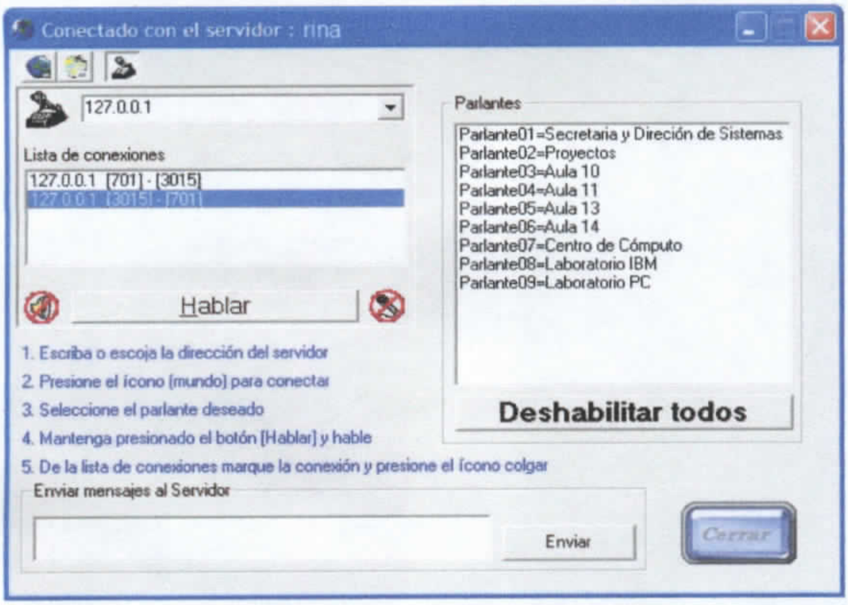


FIGURA 15. Cliente conectado al Servidor del SMDV

En la pantalla se puede observar que existen cinco pasos que los clientes deben realizar para emitir un mensaje de voz. Hasta el momento se han realizado los dos primeros pasos.