COMUNICACIÓN DIGITAL INTRATELEFÓNICA

POR

CARLOS DÍAZ GUTIÉRREZ

Tesis sometida como requisito parcial para obtener el grado de maestro en ciencias en la especialidad de computación en la Benemérita Universidad Autónoma de Puebla

SUPERVISADA POR

DRA. BÁRBARA EMMA SÁNCHEZ RINZA

PUEBLA, PUEBLA **MARZO DE 2004**

ÍNDICE

CONTENIDO **Página**

CAPÍTULO I FUNDAMENTO TEÓRICO

CAPÍTULO II DESARROLLO DE HARDWARE

CAPÍTULO III DESARROLLO DE SOFTWARE

ÍNDICE FIGURAS

ÍNDICE TABLAS

OBJETIVO GENERAL.

Elaborar un sistema de alarma láser, conjuntamente con un administrador automático de mensajes digitales locales y/o remotos, para el envío de información y recepción de órdenes, logrando aumentar la seguridad en áreas cerradas tales como casas u oficinas, y al mismo tiempo disminuir la inseguridad así como mejorar el tiempo de respuesta ante diversos eventos.

Para lograr tal fin se emplea el uso de la tecnología cuya fusión y evolución está cambiando continuamente el modo de vida y costumbres de la sociedad de nuestra época: las telecomunicaciones y la informática.

OBJETIVOS ESPECIFICOS.

- Realizar una interacción de dispositivos periféricos con la tecnología de comunicaciones.
- Administrar el envío y recepción de mensajes digitales.
- Vigilar una vivienda cuando los integrantes no se encuentran.
- Generar reportes y transmitirlos telefónicamente de forma audible o vía Internet.

JUSTIFICACIÓN.

Debido a la gran inseguridad que presenta el país, el creciente desarrollo en avances tecnológicos y al incremento de hogares que poseen un equipo de cómputo, el sistema se convierte en una alternativa factible que combina dichos elementos.

El equipo de cómputo en la mayoría de los casos es empleado para hacer tareas administrativas, y el teléfono para realizar/recibir llamadas, sin embargo estos dos elementos se pueden convertir en un sistema de vigilancia cuando los integrantes del hogar se encuentran ausentes, aprovechando de esta forma los recursos con que cuentan y evitando un pago adicional mayor al contratar un sistema de monitoreo automático.

Muchas veces al no encontrarse nadie en el hogar, los integrantes se plantean una serie de preguntas ¿Alguien me hablaría por teléfono?, ¿Quién tocaría mi puerta?, y posiblemente surgirían una serie de más interrogantes; la presente investigación ayuda a los usuarios a conocer datos referentes a estas incógnitas.

RESUMEN.

El estudio se sustenta en un conjunto de elementos cada vez más comunes en el hogar, y que debido a la constante inseguridad que presenta el país, pueden emplearse como medio de comunicación inmediata para brindar información a los integrantes de un hogar durante su ausencia.

Para lograr tal fin, el proyecto se enfoca en un sistema digital encargado de una interacción en tiempo real de dispositivos tales como computadora, cámaras web, MODEM, línea telefónica, tarjeta DTMF, Internet y bases de datos. Además se incorpora una alarma económica compuesta de un rayo láser y como sensor óptico una cámara web. Lo relevante es que al aplicar a la luz procesamiento digital se puede detectar la interrupción del láser, delatando así a personas no deseadas y activando la alarma local o remota.

Asimismo se cuenta con un administrador de datos encargado de almacenar diferentes actividades, incluyendo el envío de información a un e-mail y de teléfono a teléfono mediante un convertidor de texto a voz.

También se le incorpora al sistema un decodificador DTMF (Dual Tone Multi Frecuency) con la finalidad de ordenar procesos remotamente, así, es posible indicarle a la computadora mediante la pulsación de tonos vía teléfono digital la ejecución de tareas específicas.

El decodificador DTMF también tiene el propósito de ayudar a personas con problemas de habla, para ello se incorpora un módulo encargado de convertir cada pulsación del teclado telefónico en texto, dando lugar a un editor remoto que permite la intercomunicación telefónica.

Palabras clave: Sensor óptico, Procesamiento digital, Interacción en tiempo real, DTMF, Internet, Editor digital remoto.

INTRODUCCIÓN GENERAL.

Las telecomunicaciones y la informática, son algunas de las áreas de mayor y más rápido desarrollo en cuestión tecnológica, esto es debido a la creciente demanda de la sociedad contemporánea que exige mayores beneficios de las mismas, principalmente las grandes ciudades que requieren sistemas de comunicación modernos para satisfacer las necesidades existentes.

Actualmente se ha incrementado la inseguridad además de haberse acrecentado considerablemente la información en todos los ámbitos, por tal motivo se ve la necesidad de modernizar los sistemas de comunicación con tecnología digital, logrando así aumentar su capacidad y tiempo de respuesta ante la diversidad de sucesos que ocurren en todo momento.

De esta manera surge la propuesta de mejorar el aprovechamiento de recursos digitales con los que cada vez más personas cuentan, creando así un sistema articulado que administre diversas tareas.

Para lograr tal fin la computadora se integra como un dispositivo de seguridad apoyado de la línea telefónica, una cámara web y un láser; éste último cubriendo mediante reflexión puntos críticos o vulnerables como son puertas y ventanas; dando origen a una alarma eficaz y económica.

Además se cuenta con una serie de opciones que los sistemas actuales no integran, y que pueden activarse en forma local o remota desde cualquier teléfono digital, como es:

- Activar/desactivar seguridad por láser.
- Activar/desactivar contestador automático con la opción de guardar mensajes.
- Generación de llamadas inteligentes (determinado día/hora).
- Integración de un administrador de uso múltiple (Datos personales, Teléfonos, Direcciones de E-Mail, Programación de mensajes telefónicos, Reportes llamadas realizadas, Reportes llamadas recibidas).
- Monitoreo de la conversación telefónica actual a través de la tarjeta de sonido.
- Identificador de llamadas por MODEM.
- Clasificación de llamadas entrantes en: a) Aceptadas, b) Rechazadas, c) Normales.
- Alerta personalizada con voz o ring de llamadas detectadas, con la finalidad de saber quien llama antes de contestar.

- Mensajes de bienvenida o música personalizada por cada llamada contestada.
- Circuito DTMF para detectar la pulsación de tonos telefónicos.
- Editor digital telefónico encargado de ayudar a personas con problemas de habla o simplemente donde no se desea emplear la voz.
- Envío de llamadas grabadas a una dirección de correo u otro teléfono.
- Fotografiar las personas que tocan el timbre.
- Enviar las fotografías de personas que tocan el timbre a una dirección de correo.
- Verificar si existen nuevos e-mail en la cuenta de correo.

Con la integración de las opciones anteriormente escritas se presenta un sistema que combina diversas áreas de la informática y las telecomunicaciones, ayudando a conformar un innovador medio digital de comunicación que mantiene informados telefónicamente y por Internet a los integrantes de un hogar.

CAPÍTULO I

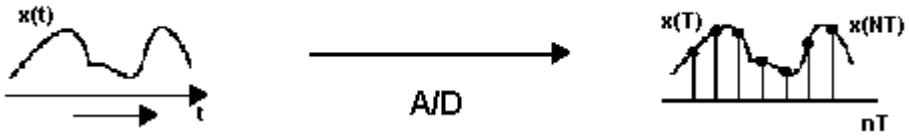
FUNDAMENTO TEÓRICO

1.1 INTRODUCCIÓN.

La mayor parte de lo que percibimos del mundo son fenómenos que existen en el tiempo, los mensajes están asociados a variables físicas (la presión en el oído, ondas luminosas en la vista) que pueden ser interpretadas como funciones reales de variable real $D = x(t)$. El tiempo es continuo y las funciones son continuas. A estas señales se les llama señales analógicas.

Normalmente se imponen restricciones para simplificar el desarrollo y para no afectar las conclusiones. Por lo tanto las funciones se tratan como suaves (derivables) y tienen una cantidad finita de energía.

Las computadoras no pueden trabajar directamente con señales analógicas[1] o continuas. Es necesario transformarlas en discretas mediante un proceso que consiste en tomar los valores de la función en diferentes valores del tiempo, como se muestra en la figura 1.1.1.



Donde:

t, T – Intervalo de tiempo.

t – Tiempo máximo del intervalo.

A/D – Analógico a Digital.

x(NT) – Valores enteros en el tiempo

nT – Frecuencia que puede representarse en segundos.

Figura 1.1.1 Convertidor analógico a digital.

Así se transforma una función real en una sucesión de números reales como se muestra en la función 1.1.1:

$$\{x(nT)\} = x(T), x2(T), \ldots, x(nT) \qquad (1.1.1)$$

Donde:

x(nT) – Función real.

x(T),x2(T),x(nT) – Sucesión de números reales.

La variable x(nT) se transforma en un número entero de modo que se puede almacenar en un número finito de bits.

El problema consiste ahora en decidir cual debe ser el intervalo *T* que se elige de modo que no se pierdan las características esenciales de la señal.

Para representar una señal discreta de longitud N se necesita el valor actual y los N-1 anteriores. Un delay (operador que retrasa la señal en un tiempo sin modificarla) es la topología natural para implementar esta descomposición.

Uno de los objetivos del procesamiento digital es encontrar la dimensión del espacio de reconstrucción que cuantifica apropiadamente las características de la señal. El tamaño de este espacio determina la longitud *N* de una ventana de tiempo que se desliza sobre toda la serie. Tamaño que corresponde a la dimensión del espacio de reconstrucción.

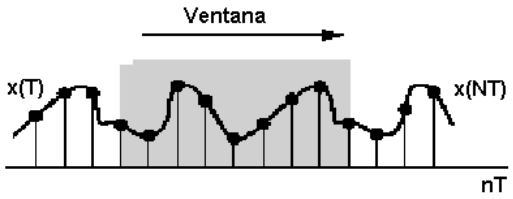


Figura 1.1.2 Ventana de longitud N.

Las técnicas de modulación están basadas en el hecho de que a través de los medios de comunicación, como pueden ser los canales telefónicos, se pueden transmitir de una forma más fiable señales analógicas que digitales. Por este motivo, cuando se va a transmitir una información digital, lo que se transmite es una señal analógica (llamada portadora) a la cual se le modifica una de sus características de acuerdo con la información binaria que se pretende transmitir. La señal portadora (carrier) es normalmente una onda senoidal, la cual está definida por tres características: frecuencia, amplitud máxima y fase. Eso quiere decir si se conocen estas tres características, en cualquier momento se puede saber el estado de la señal aplicando una fórmula matemática como se muestra la función 1.1.2.:

$$a = A * Sen(2 * f * t + \theta) \qquad (1.1.2)$$

donde:

a = valor instantáneo de la tensión en el tiempo t

A = amplitud máxima

f = frecuencia

θ = fase

t = tiempo

Si se transmite la señal portadora sin ninguna modificación, se esta transmitiendo una señal senoidal constante, la cual no transporta ninguna información. Sin embargo se puede transmitir una señal de frecuencia y fase constante, pero enviando dos amplitudes distintas, una para representar las informaciones 0 y otra para representar las informaciones 1. De la misma forma, se puede enviar una señal de amplitud y fase constantes, pero con dos frecuencias; o una señal con amplitud y frecuencia constantes, pero modificando su fase. Estos simples procesos son los que se conoce con el nombre de modulación. Dicho de otra forma, modular quiere decir modificar las características de una señal llamada portadora, de forma que contenga la información que se pretende transmitir.

Los tres sistemas básicos de modulación son los siguientes:

- Modulación de amplitud.
- Modulación de frecuencia.
- Modulación de fase.

Modulación de amplitud. Se refiere al método mediante el cual se modifica la amplitud de la señal portadora de acuerdo a la información binaria que se pretende transmitir. El método más simple de modulación de amplitud consiste en enviar una amplitud cero para representar el valor binario 0 y una amplitud determinada distinta de cero para representar el valor binario 1. También se puede transmitir una amplitud determinada para representar los valores 0 y otra amplitud distinta para representar los valores 1, ambas distintas de cero. A este sistema de modulación también se le conoce con el nombre de ASK (Amplitude-Shift Keying - Modulación por Salto de Amplitud).

La modulación de amplitud pura se emplea muy poco para transmitir datos, y si se hace, se utiliza para muy bajas velocidades de transmisión, ya que es muy susceptible a las interferencias de la línea.

Modulación de frecuencia. Con este método se modifica la frecuencia de la señal portadora de acuerdo con la información binaria que se pretende transmitir. Con este sistema se mantiene la fase y la amplitud de la señal constante y se envía una frecuencia determinada para representar el valor binario 0 y otra frecuencia distinta para representar el valor binario 1. Este salto de frecuencias también se le conoce como FSK (Frecuency-Shift Keying - Modulación por Salto de Frecuencia). La modulación se suele utilizarse para velocidades iguales o inferiores a 1200 bps.

Modulación de fase. También conocida como PSK (Phase-Shift Keying - Modulación por Salto de Fase) consiste en mantener la frecuencia y amplitud de la señal constante y modificar la fase más o menos grados dependiendo de la información binaria a transmitir.

1.2 ÁREAS DE ESTUDIO.

Durante la indagación es necesario definir el campo de búsqueda en el que se ubica el trabajo, todo esto con la finalidad de delimitar las áreas de estudio para centrarse en puntos específicos referentes a lo que representa el problema en sí. La investigación nos lleva a tratar como puntos principales los siguientes temas:

- Telefonía fija alambrica digital.
- Puerto paralelo.
- Puerto USB.
- MODEM.
- Decodificador de tonos telefónicos.
- Protocolos de correo electrónico.
- Programación visual.
- Sistemas de tiempo real.
- Principios de procesamiento digital.

Cada uno de los temas son tratados a continuación, mostrando un panorama general pero que sin embargo es de gran importancia para el desarrollo de la aplicación.

1.2.1 TELEFONÍA FIJA DIGITAL.

El teléfono es un aparato que se conecta al exterior con un par de alambres, consiste de un microteléfono y una base con un dispositivo de señalización que incluye teclado, el teléfono se encarga de transmitir señales útiles en la comunicación instantánea y remota[2] como sonidos, signos gráficos, fotografías e imágenes de televisión. Inicialmente dedicado a la transmisión de conversaciones entre dos interlocutores, el teléfono amplió poco a poco su espectro de acción mediante la conexión a diversos dispositivos, como las computadoras y otros procesadores de señales, capaces de cifrar y traducir mensajes complejos a través de líneas telefónicas. En consecuencia, se convirtió desde la segunda mitad del siglo XX en un elemento primordial dentro de los sistemas de telecomunicación.

Los tipos de señales eléctricas utilizadas son analógicas y digitales[3]. Las señales analógicas se caracterizan por tener algunos aspectos continuos y variables de la corriente eléctrica que representa la información, una señalización digital se realiza mediante pulsos estandarizados que representan información, empleando el código binario (0 y 1). 0 con la ausencia de un pulso y 1 con un pulso. La mayoría de los dispositivos electrónicos trabajan con señales digitales. Las señales se transmiten en la forma de un conjunto de bits.

El microteléfono contiene transductores electroacústicos[4], el audífono o receptor y el micrófono o transmisor. Contiene también un circuito de efecto local que permite retroalimentar hacia el receptor parte de la energía que se transmite; se entiende por efecto local al hecho de que la persona que transmite se escucha su propia voz en su mismo receptor. El nivel de este efecto se debe controlar. Cuando este es alto, la reacción natural humana hace que la persona baje el nivel de la voz. Así que regulándolo, los niveles del que transmite a su vez se pueden regular. Si se retroalimenta demasiado el nivel del receptor, se reduce el nivel de salida del transmisor como resultado de la disminución del nivel de voz de la persona, reduciéndose de esta manera el nivel o volumen de voz en el receptor distante; por lo tanto, se deteriora la operación del sistema fuente y destino.

En la figura 1.2.1.1 se muestra el diagrama esquemático de un teléfono sencillo. El estado de los switches corresponde a la condición del auricular sobre el teléfono (colgado). En el diagrama se identifican las siguientes partes:

- Timbre (B)
- Switch Hook (HS1, HS2)
- Switch de marcación (Di)
- Anti-Tinkle y speech muting (Ds)
- Híbrido (L)
- Micrófono (T)
- Bocina (R)

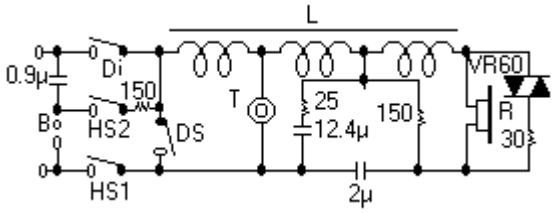


Figura 1.2.1.1 Diagrama del Teléfono.

La idea original del teléfono es muy simple, y ha perdurado hasta nuestros días. Lo único que ha cambiado con el tiempo, son los materiales de fabricación del aparato, la forma en que cada una de sus partes realiza su trabajo y la incorporación de ciertas facilidades que le brindan comodidad al usuario.

En seguida se ve la función de cada una de estas partes.

Timbre. La función del timbre es avisar que una llamada está entrando. El sonido que produce debe ser lo suficientemente fuerte, para ser escuchado por toda la casa o lugar donde se encuentre el teléfono.

El timbre utilizado en los primeros teléfonos esta compuesto por un imán permanente, dos bobinas, dos campanas y un martillo.

Este sistema fue patentado en 1878 por el señor Watson (ayudante de Bell, inventor del teléfono)

Switch Hook. La función del switch hook es desconectar y conectar el teléfono a la línea, de acuerdo con la posición del auricular. Cuando el auricular está sobre el teléfono, el switch hook, formado por HS1 y HS2, se encuentra abierto y aísla los circuitos del teléfono (excepto el timbre) de la línea. Cuando el auricular se levanta, HS1 y HS2 se cierran y conectan los circuitos del teléfono a la línea. Al cerrarse este switch (auricular levantado), se cierran también el circuito eléctrico formado por la batería de 48 VDC de la central telefónica, la resistencia limitadora de corriente, la línea y el circuito de teléfono. Este último tiene una resistencia DC variable dentro de cierto rango.

La central telefónica determina el momento en que un usuario levanta el auricular; para lograrlo, mide la intensidad de la corriente DC por medio de un detector. La intensidad de la corriente DC cuando el auricular está levantado, puede variar entre 20 y 120 miliamperios; esto depende de la resistencia de la línea y la del propio teléfono.

Switch de marcación. Este circuito especifica a la central el número telefónico al cual desea conectarse el usuario.

El circuito de marcación está compuesto por los switches Di y Ds, como se muestra en la figura 1.2.1.1 en los teléfonos de disco, estos switches se manejan por medio de un mecanismo. Si el disco se encuentra girando, el switch Ds no se cerrará; sólo lo hará hasta que aquel deje de girar. Esto se hace para evitar que la señal de la marcación se escuche en la bocina.

El switch Di se encarga de efectuar la marcación cuando el disco regresa. Durante este lapso, Di se abre y se cierra un número de veces igual al del dígito marcado. Al intervalo en que el switch

permanece abierto, se le conoce como *intervalo de Break*; y al intervalo en que el switch se encuentra cerrado, se le denomina *intervalo de make* (figura 1.2.1.2).

Esta clase de marcación, llamada *marcación por pulsos*, fue diseñada para funcionar como una planta de conmutación electromecánica con un promedio límite de detección de 10 pulsos (break) por segundo.

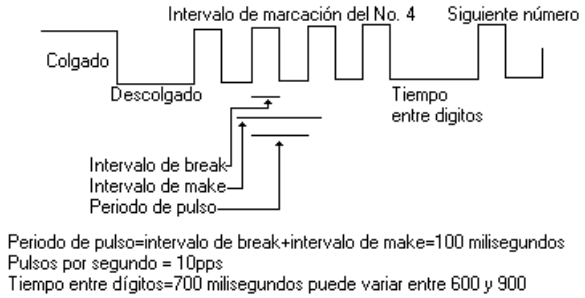


Figura 1.2.1.2 Intervalos en la marcación de números.

Anti-Tinkle y Speech Muting. Esta función se cumple por medio de la resistencia de 150 ohmios y el switch Ds, los cuales se muestran en la figura 1.2.1.1 Durante la marcación, como ya se menciono Ds, permanece cerrado hasta que el disco se pone en reposo. Este switch realiza un muting, para evitar que la marcación se escuche; o para evitar algo peor: que se dañe la bocina.

Híbrido. En la figura 1.2.1.1 se muestra el híbrido, que corresponde al transformador marcado con la letra L, básicamente es un transformador de audio.

Este circuito cumple varias funciones: sirve como interfaz entre un circuito de cuatro alambres. El auricular telefónico es un circuito de cuatro alambres, de los cuales dos son para el micrófono y dos para la bocina; en tanto, la línea telefónica tiene dos alambres.

Pero eso no es todo, ya que este circuito también se encarga de producir el *sidetone*, Y gracias a esto, se puede escuchar en la bocina, con un volumen adecuado, lo que se está transmitiendo por el micrófono.

Por medio de este circuito, también se puede manejar la impedancia del teléfono con la de la línea. Así se evitan reflexiones (ecos), porque toda la energía que llegue al teléfono será disipada.

Micrófono. Este dispositivo convierte la energía acústica en una señal eléctrica de voltaje, permitiendo el envío de la voz a través de la línea telefónica.

Para el desarrollo de este proyecto se emplean señales digitales provocadas al pulsar algún digito del teclado.

El empleo en la actualidad de señales digitales es debido a los avances más recientes de la comunicación que tienden a favorecer un mayor grado de competencia en el campo de la comunicación de datos.

1.2.2 PUERTO PARALELO.

Las computadoras han estandarizado un tipo de interfaz para la comunicación con la impresora, conocida normalmente como *centronics*[5]. Esta interfaz es capaz de enviar caracteres a la impresora de forma paralelo. Cada carácter está codificado en un byte, del cual cada bit se transmite por un pin diferente. Existen otros pines que se conectan a la computadora e impresora, que sirven para intercambiar información de control y de estado, a fin de implementar un sencillo protocolo.

Para realizar esta interfaz, la computadora dispone de elementos hardware específicos, denominados puertos de impresora o puertos paralelo.

En una computadora pueden instalarse varios de estos puertos que se distinguen entre sí por los nombres LPT1, LPT2 y LPT3 (en algunos casos también LPT4). Todos ellos son idénticos, salvo que tienen asignadas diferentes direcciones en el mapa de entrada/salida.

Cada uno de estos puertos ocupa tres direcciones[6] del mapa de E/S:

- *Puerto de Datos:* de 8 bits, es donde la CPU escribe los datos que se envían a la impresora (caracteres).
- *Puerto de Estado:* Registro de 8 bits de donde la CPU puede conocer diversos aspectos del estado de la impresora (apagada, sin papel). Se usan sólo 5 de los 8 bits.
- *Puerto de Control:* Registro de 8 bits donde la CPU puede escribir diversas señales que reconoce la impresora (validación de datos, inicialización) se usan 4 bits.

Las direcciones del mapa de memoria de E/S en las que se suelen colocar los puertos de impresora son:

Direcciones de LPTn			
PUERTO	DATO	STATUS	CONTROL
LPT1	378H	379H	37AH
LPT2	278H	279H	27AH
LPT3	3BCH	2BDH	3BEH

Tabla 1.2.2.1 Direcciones del puerto paralelo.

Señales de la interfaz. El puerto paralelo, utilizado en general para el control de la impresión, maneja las señales que se muestran en la tabla 1.2.2.2. De todas ellas, las que normalmente intervienen en el protocolo de comunicación entre la computadora y la impresora son BUSY y STROBE#. Algunas de las impresoras pueden no utilizar alguna de las señales descritas en la tabla.

PUERTO	NOMBRE	TIPO	Pin DB25	DESCRIPCIÓN
DATOS	D0…D7	S	2-9	8 terminales de datos. Desde D0 a D7.
ESTADO	BUSY	E	11	Un nivel alto indica que la impresora está ocupada y no puede recibir datos nuevos. También se pone a 1 en situaciones de error
	ACK#	E	10	Un nivel bajo indica que la impresora ha recibido un dato y está disponible para recibir uno nuevo.
	POUT	E	12	Un nivel alto indica que la impresora no tiene papel.
	SLCT IN	E	13	Un nivel alto indica que la impresora está *on-line*.
	ERROR#	E	15	Un nivel bajo indica que se ha producido un error en la impresora. No hay papel, mal funcionamiento.
CONTROL	SELECT#	S	17	Activa a nivel bajo. Indica que se ha seleccionado la impresora.
	INIT#	S	16	A nivel bajo, envía un RESET a la impresora.
	AUTOFD#	S	14	A nivel bajo, la impresora se encarga de hacer un salto de línea al recibir el carácter "retorno de carro"
	STROBE#	S	1	Validación de datos. Cuando la impresora detecta un nivel bajo, acepta el dato.

* El carácter # indica que la señal es activada a nivel bajo.

Tabla 1.2.2.2 Señales del puerto paralelo.

1.2.3 PUERTO USB.

Como resultado de dotar a la computadora de un bus de alta velocidad que ofreciera las características ideales de universalidad, facilidad de conexión/desconexión y sobre todo, que consumiese pocos recursos, Intel y otros lideres de la industria diseñaron el denominado puerto *USB Universal Serial Bus*[7], que como su nombre lo indica, es un bus serie, bidireccional y de bajo costo; diseñado como una extensión en la arquitectura estándar del PC, orientado

principalmente en la integración de periféricos, que aparecen como un solo puerto en lo que se refiere a utilización de recursos.

En sus comienzos los interesados en esta tecnología se agruparon en un foro, el USB Implementers Forum., USB-IF, que agrupa a más de 400 compañías (Compaq, Hewlett-Packard, Intel, Lucent Technologies, Microsoft, NEC y Philips entre otras) han publicado diversas revisiones de la norma:

USB 0.9: Primer borrador, publicado en Noviembre de 1995.

USB 1.0: Publicada en 1996 establece dos tipos de conexión. La primera, denominada velocidad baja (low speed), ofrece 1.5 Mbps. La segunda, denominada velocidad completa ("Full speed"), es de 12 Mbps.

USB 1.1: Publicada en 1998, añade detalles y precisiones a la norma inicial; es el estándar mínimo que debe cumplir un dispositivo USB.

USB 2.0: Su versión final fue publicada en Abril del 2000; es una extensión de la norma compatible con las anteriores. Permite velocidades de hasta 480 Mbps, denominada alta velocidad ("High speed").

El primer ordenador que incluyó un puerto *USB* de forma estándar fue el iMac de Apple, presentado en marzo de 1998, utilizaba esta conexión para el teclado y el ratón. Otros fabricantes de computadoras lo comenzaron a utilizar cuando Microsoft introdujo los controladores correspondientes en la versión OSR 2.1 de Windows 95. Fue a partir de Windows 95 cuando los sistemas incorporan de forma estándar soporte para este bus; en el ámbito de servidores la incorporación se produjo en Windows 2000.

Los primeros dispositivos que empezaron a utilizar masivamente este tipo de conexión fueron las cámaras de video-conferencia, aunque actualmente pueden encontrarse todo tipo de dispositivos para este estándar de conexión.

El protocolo de comunicación utilizado es de testigo, que guarda cierta similitud con el sistema Token-Ring de IBM. Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultanea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido. Existen cuatro tipos de paquetes distintos: *Token; Datos; Handshake*, y *Especial*; el máximo de datos por paquete es de 8; 16; 32 y 64 bytes. Se utiliza un sistema de detección y corrección de errores bastante robusto tipo CRC ("Cyclical Redundancy Check").

El funcionamiento está centrado en el host, todas las transacciones se originan en él; es el controlador host el que decide todas las acciones, incluyendo el número asignado a cada dispositivo (esta asignación es realizada automáticamente por el controlador "host" cada vez que se inicia el sistema o se añade, o elimina, un nuevo dispositivo en el bus), su ancho de banda, etc. Cuando se detecta un nuevo dispositivo es el host el encargado de cargar los drivers oportunos sin necesidad de intervención por el usuario.

El sistema utiliza cuatro tipos de transacciones que resuelven todas las posibles situaciones de comunicación[8]; cada transacción utiliza un mínimo de tres paquetes, el primero es siempre un *Token*, que avisa al dispositivo que puede iniciar la transmisión.

- *Transferencia de control* ("Control transfer"): Ocurre cuando un dispositivo se conecta por primera vez; en este momento el controlador de host envía un paquete "Token" al periférico notificándosele el número que le ha asignado.
- *Transferencia de pila de datos* ("Build data transfer"): Este proceso se utiliza para enviar gran cantidad de datos en una sola vez. Es útil para dispositivos que tienen que enviar gran cantidad de datos cada vez, como escáneres o máquinas de fotografía digital.
- *Transferencia por interrupción* ("Interrupt data transfer"): Este proceso se utiliza cuando se solicita enviar información por el bus en una sola dirección (de la función al host).
- *Transferencia de datos isócrona* ("Isochronous data transfer"): Este proceso se utiliza cuando es necesario enviar datos en tiempo real. Los datos son enviados con una cadena precisa ajustada a un reloj, de modo que la transmisión es a velocidad constante.

Las comunicaciones asíncronas ponen más énfasis en garantizar el envío de datos, y menos en su temporización cuando llegan; por su parte las comunicaciones isócronas son lo contrario, ponen más énfasis en la oportunidad de la transmisión que en la velocidad. Esta sincronización es importante en situaciones como la reproducción de video, donde no debe existir desfase entre las señales de video y audio.

El cable de bus USB es de 4 hilos, y comprende líneas de señal (datos) y alimentación, con lo que las funciones pueden utilizar un único cable como se muestra en la siguiente tabla.

Pin	Nombre	Descripción	Color
1	VBUS	+5 V. CC	Rojo
2	D-	Data -	Azul
3	D+	Data +	Amarillo
4	GND	Tierra	Verde

Tabla 1.2.3.1 Líneas de señal del puerto USB.

Las principales ventajas del Bus Serie Universal son:

1. Un único conector para todos los periféricos compatibles USB, hasta 127 dispositivos se pueden conectar en serie.
2. La alta velocidad de transferencia del sistema, que ofrece 12 Mbits por segundo. Muy superior a los 115 Kbits por segundo que ofrece el actual puerto de comunicaciones serie de las computadoras.
3. Facilidades *Plug & Play* para determinar el tipo de periférico que se encuentra en uso y la posibilidad de conectar y desconectar dispositivos sin la necesidad de reiniciar la máquina.

1.2.4 MODEM.

El MODEM (modulador-demodulador) está preparado para recibir comandos AT que consiste en un juego de comandos básicos y en una extensión de comandos. El juego de comandos básicos es común en todos los MODEMS Hayes y compatibles; sin embargo el juego de comandos extendidos sólo es aplicable a aquellos MODEMS que dispongan de esas características o modos de operación. Eso quiere decir que si se dispone de un determinado software de comunicaciones y se quiere estar seguro que es capaz de aprovechar al máximo las características del MODEM, la única forma de tener la certeza es comprobando que soporta el modelo específico de MODEM que se esta usando. En cualquier otro caso, la única seguridad es que el software compatible Hayes hará uso del juego de comandos básicos.

Una vez ejecutado el software y antes de conectarse a línea telefónica (modo "offline") o una vez conectado a línea y tras haber recibido la secuencia de escape (+ + +, modo "online"). Los comandos pueden ser enviados al módem a través de una computadora o a través de un programa de comunicaciones[9].

Los comandos deben empezar con el prefijo **AT** seguidos por la letra del comando y terminados con la tecla **ENTER**. En una misma cadena se pueden enlazar varios comandos en cuyo caso se irán añadiendo las letras de los mismos sin necesidad de ir introduciendo los caracteres **AT.** Los espacios están permitidos en la cadena de comandos, para así incrementar la legibilidad de la cadena, pero son ignorados por el módem. Todos los comandos deben ser escritos en letra minúscula o mayúscula pero no ambos a la vez.

Un comando enviado sin ningún parámetro, es considerado como un comando con parámetro **"0"**.

Ejemplo: ATL[ENTER]

Este comando ajusta al mínimo el volumen del módem.

La siguiente tabla muestra los principales comandos estándar:

COMANDO	FUNCIÓN	COMANDO	FUNCIÓN
A	Responder llamada entrante	M0	Altavoz siempre apagado
A/	Repite el último comando. No necesita ir precedido por AT	M1	Altavoz encendido hasta detección de portadora.
P	Marcar por pulsos	Sr?	Leer registro r
Sr=n	Colocar n en el registro r	T	Marcar por tonos
E0	Eco de comandos no activo	E1	Eco de comandos activo
+ + +	Pasa a modo comandos en línea	H0	Se desconecta de línea
H1	MODEM se coloca en ocupado	L0	Volumen altavoz mínimo
L1	Volumen altavoz bajo	L2	Volumen altavoz medio
L3	Volumen altavoz alto		

Tabla 1.2.4.1 Principales comandos AT.

Algunos registros pueden cambiar su valor a través de los comandos AT[10]. Para que un cambio de valor en un registro tenga efecto, se tendrá que resetear el módem. La siguiente tabla muestra cada uno de los registros que pueden cambiar su valor:

REGISTRO	FUNCIÓN	RANGO/UNIDADES
S0	Número de rings antes de contestar	0-255 rings
S1	Número de rings recibidos	0-255 rings
S2	Código del carácter de escape	0-255/ASCII
S3	Carácter de retorno de carro	0-127/ASCII
S4	Carácter de salto de línea	0-127/ASCII
S5	Carácter de borrado	0-255/ASCII
S6	Tiempo de espera de tono de línea	2-255 segundos
S7	Tiempo de espera de portadora	1-255 segundos
S8	Tiempo de la pausa de la coma	0-255 segundos
S9	Tiempote detección de portadora	1-255/0.1 segs
S10	Tiempo de perdida de portadora	1-255/0.1 segs
S11	Velocidad de marcación por tonos	50-255/0.001 segs
S12	Tiempo de detección de carácter de escape	0-255/0.02 segs
S14	Opciones de estado del módem	Mapeado de bits
S18	Contador de test del módem	0-255 segs
S22	Altavoz y respuesta	Mapeado de bits
S29	Tiempo de duración del Flash	0-255 (x10 msegs)
S30	Temporizador de desconexión	0-255 (x10segs)
S32	Carácter XON	0-255 ASCII
S33	Carácter XOFF	0-255 ASCII
S37	Velocidad de conexión en línea	
S38	Retardo después de colgar	0-255 segundos

Tabla 1.2.4.2 Registros del MODEM.

En la siguiente tabla se muestran los códigos de resultado enviados por el módem como respuesta a la entrada de comandos AT o diferentes estados del módem[11]. En la columna izquierda se presentan en formato alfanumérico y en la columna de la derecha en formato numérico, la presentación dependerá del tipo de invocación que se haya indicado.

ALFANUMÉRICO	NUMÉRICO	ALFANUMÉRICO	NUMÉRICO
OK	0	CONNECT	1
RING	2	NO CARRIER	3
ERROR	4	CONNECT1200	5
NO DIALTONE	6	BUSY	7
NO ANSWER	8	CONNECT0600	9
CONNECT2400	10	CONNECT4800	11
CONNECT9600	12	CONNECT7200	13
CONNECT12000	14	CONNECT14400	15
CONNECT19200	16	CONNECT38400	17
CONNECT57600	18	CONNECT115200	19

Tabla 1.2.4.3 Valores devueltos por el MODEM.

1.2.5 DECODIFICADOR DE TONOS TELEFÓNICOS.

El servicio telefónico es un medio que da la oportunidad de realizar a distancia diversas tareas. La red telefónica que emplea señales por tonos es la más utilizada actualmente ya que proporciona un marcado más rápido del número deseado y además la mayoría de las centrales telefónicas son digitales.

El marcado por tonos tiene su fundamento en la señal de DTMF (Dual Tone Multi Frecuency), es decir Multi Frecuencia de Dos Tonos, esto quiere decir que el digito marcado esta compuesto por dos frecuencias, una frecuencia alta y una frecuencia baja, que se manda simultáneamente a través de la línea telefónica[12].

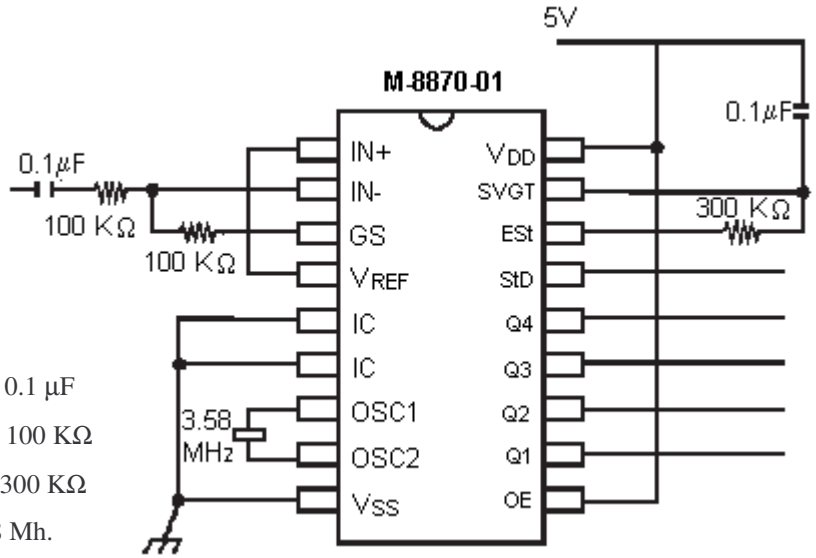
Los dígitos DTMF mostrados en la figura 1.2.5.1 son un estándar internacional de codificación por tonos para ser utilizados en líneas telefónicas convencionales.

El decodificador se encarga del reconocimiento de los pares de frecuencia, y su salida de validación indicando que se ha decodificado una pulsación, obteniendo su valor en las salidas correspondientes[13], el circuito se conecta al puerto paralelo mediante un DB25.

FBAJA	FALTA	TECLA	Q4	Q3	Q2	Q1
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1477	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1477	9	1	0	0	1
941	1336	0	1	0	1	0
941	1209	S	1	0	1	1
941	1477	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

Figura 1.2.5.1 Frecuencias estándar de tonos telefónicos.

El Circuito Integrado M-8870-01 decodifica 16 tonos telefónicos que son suficientes para el desarrollo de este proyecto[14].



2 – Capacitores 0.1 µF
2 - Resistencias 100 KΩ
1 – Resistencia 300 KΩ
1 – Cuarzo 3.58 Mh.
5V – Obtenidos del puerto paralelo

Figura 1.2.5.2 Circuito M-8870-01

1.2.6 PROTOCOLOS DE CORREO ELECTRÓNICO.

Los primeros sistemas de correo electrónico simplemente consistían en protocolos de transferencia de archivos[15], con la convención de que la primera línea de cada mensaje (es decir del archivo) contenía la dirección del destinatario.

Limitaciones de este sistema es el envío a grupos, sin notificación. En 1982 se publicaron las propuestas de correo electrónico del ARPANET como el protocolo de transmisión SMTP.

Dos años después, el CCITT elaboró su recomendación X.400, pero su excesiva complejidad, hace que no se utilice, como la mayoría de aplicaciones OSI.

La transferencia de datos se clasifica en dos agentes[16]:

- *Distribución:* utilizando para ello el protocolo SMTP (Simple Mail Transfer Protocol-Protocolo de Transferencia de Correo Simple) o SMTP extendido (ESMTP).
- *Entrega final:* que permita al usuario gestionar su correo a través de una máquina remota, utilizando los protocolos POP3 (Post Office Protocol-Protocolo de Oficinas de Correos) e IMAP (Interactive Mail Access Protocol-Protocolo de Correo de Acceso Interactivo).

El SMTP es un sencillo protocolo *cliente/servidor* en formato *ASCII*[17]. Establecida una comunicación *TCP* entre la computadora transmisora del correo, que opera como cliente, y el *puerto 25* de la computadora receptora del correo, que opera como servidor, el cliente permanece a la espera de recibir un mensaje del servidor.

El servidor *comienza por enviar una línea de texto* que proporciona su identidad *e indica si está preparado o no* para recibir correo:

a. *Si no lo está*, el cliente libera la conexión y lo intenta después.
b. *Si está dispuesto* a aceptar correo electrónico, el cliente anuncia de quién viene el mensaje, y a quién está dirigido. Si existe tal destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. Entonces el cliente envía el mensaje y el servidor acusa su recibo, si existe más correo electrónico también se envía ahora. Una vez que todo el correo ha sido intercambiado *en ambas direcciones*, se libera la conexión.

En la siguiente tabla se muestran los comandos que se pueden emplear[18,19]:

COMANDO	DESCRIPCIÓN
HELO	Identifica el remitente al destinatario.
MAIL FROM	Identifica una transacción de correo e identifica al emisor.
RCPT TO	Se utiliza para identificar un destinatario individual. Si se necesita identificar múltiples destinatarios es necesario repetir el comando.
DATA	Permite enviar una serie de líneas de texto. El tamaño máximo de de una línea es de 1000 caracteres. Cada línea va seguida de un retorno de carro y avance de línea. <CR> <LF>. La última línea debe llevar únicamente el carácter punto "." seguido de <CR> <LF>.
RSET	Aborta la transacción de correo actual.
NOOP	No operación. Indica al extremo que envíe una respuesta positiva.
QUIT	Pide al otro extremo que envíe una respuesta positiva y cierre la conexión.
VRFY	Pide al receptor que confirme que un nombre identifica a un destinatario valido.
EXPN	Pide al receptor la confirmación de una lista de correo y que devuelva los nombres de los usuarios de dicha lista.
HELP	Pide al otro extremo información sobre los comandos disponibles.
TURN	El emisor pide que se inviertan los papeles, para poder actuar como receptor. El receptor puede negarse a dicha petición.
SOML	Si el destinatario está conectado, entrega el mensaje directamente a la computadora, en caso contrario lo entrega como correo convencional.
SAML	Entrega el mensaje en el buzón del destinatario. En caso de estar conectado también lo hace a la computadora cliente.
SEND	Si el destinatario está conectado, entrega el mensaje directamente a la computadora cliente.

Tabla 1.2.6.1 Comandos SMTP.

La siguiente tabla muestra los códigos de respuesta:

COMANDO	DESCRIPCIÓN
211	Estado del sistema.
214	Mensaje de ayuda.
220	Servicio preparado.
221	Servicio del canal de transmisión cerrando.
250	Solicitud completada con éxito.
251	Usuario no local, se enviará a <dirección de reenvío>.
354	Introduzca el texto, finalice con <CR><LF>.<CR><LF>.
421	Servicio no disponible.
450	Solicitud de correo no ejecutada, servicio no disponible (buzón ocupado).
451	Acción no ejecutada, error local de procesamiento.
452	Acción no ejecutada, insuficiente espacio de almacenamiento en el sistema.
500	Error de sintaxis, comando no reconocido.
501	Error de sintaxis. Ejemplo. contestación de SMTP a ESMTP.
502	Comando no implementado.
503	Secuencia de comandos errónea.
504	Parámetro no implementado.
550	Solicitud no ejecutada, buzón no disponible.
551	Usuario no local, pruebe <dirección de reenvío>.
552	Acción de correo solicitud abortada.
553	Solicitud no realizada (error de sintaxis).
554	Fallo en la transacción.

Tabla 1.2.6.2 Códigos de respuesta SMTP.

Algunos de los inconvenientes al utilizar SMTP son:

- Algunas implementaciones más viejas de SMTP no pueden manejar mensajes *mayores de 64 Kbytes*.
- Si el cliente y el servidor tienen *temporizaciones distintas*, uno de ellos puede terminar mientras que el otro continúa trabajando, terminando inesperadamente la conexión.
- En ocasiones pueden dispararse *tormentas de correo infinitas* cuando ambos servidores mutuamente tienen una lista que incluye a la otra lista del otro servidor.

- Solución, un nuevo protocolo extendido: *SMTP extendido (ESMTP)*. Los clientes que deseen usarlo deben enviar un mensaje *EHLO*, en lugar de HELO. Si el saludo se rechaza, *código 500*, esto indica que se trata de un servidor SMTP normal y el cliente debe proceder de la manera normal.

El correo entrante en un cliente se puede realizar a través de los siguientes protocolos:

- *POP3.* Tiene comandos para que un usuario establezca una sesión (USER y PASS), la termine (QUIT), obtenga mensajes (RETR) y los borre (DELE). El protocolo mismo consiste en texto ASCII y se asemeja a SMTP. El objetivo del POP3 es obtener correo electrónico del *buzón remoto y almacenarlo en la máquina local del usuario* para su lectura posterior.
- *IMAP*. La idea en que se basa IMAP es que el servidor de correo electrónico mantenga un depósito central al que puede accederse desde cualquier máquina. Por tanto, a diferencia del POP3, *no copia el correo electrónico en la máquina personal del usuario dado que el usuario puede tener varias computadoras para consultar el correo*, y observa si sus correos han sido leídos con anterioridad.
- *DMSP* (Distributed Mail System Protocol) es parte del sistema PCMAIL *es un sistema de correo electrónico distribuido*, que permite leer y contestar estando desconectado y permite gestionar varios servidores. Al ocurrir una reconexión después, el correo electrónico se transferirá y el sistema se sincronizará.

Otras características de los agentes de transferencia son:

- Pueden incorporar *filtros o reglas* cuando llega un correo electrónico.
- Pueden *reenviar* a una dirección diferente, por ejemplo un teléfono móvil con SMS, o a otro servidor.
- Permiten generar una *contestación automática*, por ejemplo cuando estamos de vacaciones: "*Estoy de vacaciones. Regresaré el 15 de Agosto. Que tenga feliz día*".

1.2.6.1 DESCRIPCIÓN DEL PROTOCOLO POP3.

Para que un usuario pueda recibir correo mediante SMTP necesita tener una cuenta donde almacenar los mensajes que recibe. El envío de mensajes se realiza estableciendo una conexión TCP con el puerto 25 del servidor SMTP donde esta el correo del usuario, y transfiriendo después los mensajes pendientes, que se almacenarán en el correo. El servidor SMTP debe estar permanentemente en funcionamiento para que los mensajes puedan recibirse y almacenarse en cualquier instante del día.

Frecuentemente, se trabaja en computadoras que se apagan al finalizar la jornada de trabajo, y por lo tanto no resultan adecuadas para ejecutar los servidores SMTP, por no estar siempre disponibles. A esto hay que añadir el hecho de que administrar y configurar los servidores SMTP supone un trabajo, en el que la mayoría de los usuarios no están interesados. Por estos motivos se diseñó otro protocolo, el POP3, que permite a los usuarios consultar el correo previamente recibido y almacenado en el buzón, desde una computadora de trabajo habitual. Para leer los mensajes, el cliente POP3 utiliza una conexión TCP con el puerto 110 del servidor[20]. Este protocolo *no sirve para enviar mensajes* a otros usuarios, sino únicamente para *consultar* los mensajes previamente *recibidos* y ya almacenados en el buzón del usuario.

Las órdenes que envía el cliente pueden estar en mayúsculas o minúsculas, constan de una palabra reservada de 3 o 4 letras, que según los casos va seguida o no, por una serie de argumentos. Cuando se utilizan argumentos van separados por un único espacio en blanco. Todas las órdenes terminan con la secuencia chr(13) + chr(10) o por CR+LF.

Las respuestas del servidor consisten en un indicador de estado, al que posiblemente sigue información adicional. Todas las respuestas terminan con CR+LF y los indicadores de estado deben estar en mayúsculas. Actualmente, hay dos indicadores de estado: uno positivo (“+OK”) y otro negativo (“-ERR”).

Las respuestas a algunas ordenes son multilínea. En estos casos, tras enviar la primera línea que terminará con la secuencia CR+LF pueden recibirse líneas adicionales, siempre acabadas con CR+LF. Después de enviar todas las líneas de respuesta, el servidor envía una última línea, formada por un byte de terminación (el código decimal 46, “.”) y CR+LF. Si cualquier línea de la respuesta empieza con un byte de terminación, el servidor aplica relleno de carácter (“byte-stuffing”) y lo envía por duplicado. Al examinar una respuesta multilínea, el cliente analizará el primer byte de la misma, y si se trata de un byte de terminación, comprobará si va seguido de la

secuencia CR+LF. En caso afirmativo se trata del final de la respuesta multilínea, y en otro caso hay que eliminar el primer carácter "." que el cliente había duplicado.

El funcionamiento del protocolo es el siguiente:

a) Inicio de conexión. Al establecerse la conexión el servidor devuelve un saludo, que puede ser cualquier secuencia positiva, por ejemplo:

S: + OK POP3 server ready CR+LF

b) Fase de autorización. A continuación, el cliente realizará la autorización enviando un nombre de usuario y una contraseña, por ejemplo:

C: USER nombre CR+LF

S: -ERR/+OK ... CR+LF

Si la respuesta del servidor es +OK:

C: PASS string CR+LF

S: -ERR/+OK ... CR+LF

c) Fase de transacción. Una vez que el cliente se ha identificado con éxito ante el servidor la sesión pasa al estado de transacción, durante la que el cliente podrá utilizar las órdenes que se muestran a continuación[21], incluso de forma repetida. Finalmente, el cliente enviará la orden QUIT y la sesión POP3 pasará al estado de actualización.

- *STAT.* Indica el número de mensajes que hay en el buzón del usuario. Si el servidor envía una respuesta positiva consistirá en la cadena "+OK" seguida de un espacio, el número de mensajes en el buzón, un único espacio y el número de bytes consumidos del buzón. Por ejemplo:

C: STAT

S: +OK 2 320

- *LIST [msg].* Cuando se da un argumento (número de mensaje en el buzón) y el servidor proporciona una respuesta positiva, esta consta de una línea con información sobre el mensaje (como mínimo el tamaño del mensaje en bytes).

  Si no se dan argumentos y la respuesta del servidor es positiva, proporciona una línea de información por cada uno de los mensajes almacenados en el buzón. Por ejemplo:

C: LIST

S: +OK 2 messages (320 bytes)

S: 1 120

S: 2 200

S:.

...

C: LIST 3

S: -ERR no such message, only 2 messages in maildrop

- *RETR msg.* Si la respuesta es positiva, se trata de una respuesta multilínea. Después del +OK inicial, el servidor envía el mensaje correspondiente al numero solicitado, aplicando *byte-stuffing* cuando resulte necesario.

C: RETR 1

S: +OK 120 octets

S: < aquí va el mensaje>

S: .

- *DELE msg.* Indica al servidor que marque el mensaje indicado como borrado. Cualquier referencia posterior a el generará un mensaje de error. Realmente, el mensaje no se borra hasta que la sesión entra en el estado de actualización (orden QUIT), por lo que es posible recuperarlo, mientras tanto, mediante la orden RSET. Por ejemplo:

C: DELE 1

S: +OK message 1 deleted

...

C: DELE 2

S: -ERR message 2 already deleted

- *RSET.* Permite recuperar los mensajes que hayan sido borrados durante la sesión.

C: RSET

S: +OK maildrop has 2 messages

d) Fase de actualización. Ocurre cuando un cliente que estaba en la fase de transacción envía la orden QUIT.

- *QUIT.* El servidor elimina los mensajes marcados como borrados del buzón y cierra la conexión TCP. Por ejemplo:

C: QUIT

S: +OK POP3 server signing off (maildrop empty)

1.2.6.2 TRANSFERENCIA DE ARCHIVOS POR E-MAIL.

Uno de los usos más extendidos del e-mail es el envío de archivos. Desde un principio Internet se basó sobre los caracteres ASCII "puros" de 7 bits, es decir, desde el 0 al 127, que era ideal para casi todos los archivos de texto, sobre todo si estaban en inglés. Sin embargo, los archivos binarios y otros archivos de texto basados en nuestro idioma (que incluyen eñes y acentos) requieren del ASCII extendido, con caracteres de 0 a 255. Para solucionar este problema, se fueron implementando varios protocolos adicionales para enviar archivos binarios en formato de 7 bits. El más difundido es el *MIME* (Extensiones Multipropósito para Correo de Internet)[22], el mismo protocolo de identificación que utiliza la WWW.

La ventaja de enviar archivos binarios, es que no tenemos que compatibilizar formatos, pues el mensaje o archivo en binario llega tal como se generó, y en el caso de archivos de texto, el software mismo se encargará de darle formato correspondiente (ejemplo. Word). Para otros tipos de archivos, como gráficos (.jpg o .gif) o de sonido (.wav y .au), se debe tener el software adecuado para poder visualizarlo y reproducirlo, según sea el caso. Generalmente se utiliza el comando "Adjuntar" (Attach), en los softwares de Correo que lo posean. Esto es, se adiciona uno o varios archivos a un mensaje.

Formato de mensajes MIME.

- Campos de cabecera
  - MIME-Version: 1.0
  - Content-Type: tipo, subtipo y parámetros opcionales
  - Content-Transfer-Encoding: codificación
  - Content-ID: identificador único para cada parte del mensaje (opcional)
  - Content-Description: comentario asociado (opcional)
- Tipos de contenido
  - text/plain, text/enriched, text/HTML
    - Parámetro.
  - image/gif, image/jpeg, image/png, …
  - audio/basic: 8 bits, 8000 Hz, un canal
  - video/mpeg
  - application/octet-stream, application/postcript

- Message
  - un mensaje completo (ejemplo. reenviar de mensajes de error)
  - partial: un fragmento
  - external-Body: una referencia (ejemplo. FTP) a objeto externo
- Multipart: el mensaje contiene varias partes
  - mixed: independientes
  - alternative: mismo contenido en varios formatos para elegir uno
    - Ejemplo: texto/html/rtf
  - digest: colección de mensajes
  - parallel: varias partes que deben presentarse simultáneamente
    - Ej. audio y video
  - related: las partes constituyen un único documento
    - Ejemplo: página web
  - report: acuse de recibo

- Codificación
  - Mecanismos para
    - Convertir datos al formato de 7 bits del US-ASCII
    - Codificar archivos
  - Campo Content-transfer-encoding

CODIFICACIÓN	INFORMACIÓN	CODIFICADO	VIAJA
7bit	7 bits	7 bits	SMTP
quoted-printable	8 bits	7 bits	SMTP
base64	8 bits	7 bits	SMTP
8bit	8 bits	8 bits	ESMTP/8 bits
Binary	8 bits	8 bits	ESMTP/binario

Tabla 1.2.6.2.1 Tipos de codificación de archivos anexos.

- Codificación quoted-printable
  - Mayoría texto: no se codifica (7 bits, caracteres del juego US-ASCII)
  - Caracteres especiales: se transforman
    - '=' + codif. ASCII en Hexa
  - Resultado casi legible sin decodificar
    - Ejemplo
      - Texto: El camión se salió de la cañada
      - Codificación: El cami=F3n se sali=F3 de la ca=F1ada
- Codificación base 64
  - Similar / evolución uuencode
  - Emplea un subconjunto del US-ASCII de 64 caracteres
  - Grupos de 24 bits de entrada → 4 grupos de 6 bits → 4 caracteres codificados del *alfabeto base 64* → 4 ASCII → 32 bits
    - Incrementa el tamaño de los mensajes un 33%
- Grupo de menos de 24 bits: se rellena con bits 0
- Decodificación:
  - Relleno con '=' para que el número de caracteres a la salida sea múltiplo de 4
  - Se ignoran caracteres que no estén en la tabla (CR, LF, *, ...)
- Codificación base 64: alfabeto

VALOR	CÓDIGO	VALOR	CÓDIGO	VALOR	CÓDIGO	VALOR	CÓDIGO	VALOR	CÓDIGO
0	A	13	N	26	a	39	n	52	0
1	B	14	O	27	b	40	o	53	1
2	C	15	P	28	c	41	p	54	2
3	D	16	Q	29	d	42	q	55	3
4	E	17	R	30	e	43	r	56	4
5	F	18	S	31	f	44	s	57	5
6	G	19	T	32	g	45	t	58	6
7	H	20	U	33	h	46	u	59	7
8	I	21	V	34	i	47	v	60	8
9	J	22	W	35	J	48	w	61	9
10	K	23	X	36	K	49	x	62	+
11	L	24	Y	37	L	50	y	63	/
12	M	25	Z	38	M	51	z	(PAD)	=

Tabla 1.2.6.2.2 Alfabeto base 64.

- Codificación base 64
  - Esta codificación se emplea para la transmisión de archivos anexos (Attach) o simplemente en el envío de mensajes.

    A continuación de muestra un ejemplo en el que se codifica ALEX a base 64 para.

    ALEX

    Texto: A L E X

    Hex : 0x41 0x4C 0x45 0x58

    Bin: 01000001 01001100 01000101 01011000

    Codif: 010000 010100 110001 000101 010110 000000 0..0 0..0

    Decimal: 16 20 49 5 22 0 0 0

    Base-64: Q U x F W A A A

    Una vez codificada la palabra ALEX en base 64 se transmite su equivalente QUxFWAAA.

1.2.7 PROGRAMACIÓN VISUAL.

Es una herramienta de aplicaciones para Windows, que se emplea en gran medida para el diseño de una interfaz gráfica. En una aplicación el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfaz gráfica[23].

Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias, ya que no se puede decir que la programación Visual pertenezca por completo a uno de esos dos tipos de programación.

La creación de un programa bajo ambiente Visual lleva los siguientes pasos:

1. Creación de una interfaz de usuario. Esta interfaz será la principal vía de comunicación hombre - máquina, tanto para la salida de datos como para la entrada. Es necesario partir de una ventana también conocida como Formulario, a la que se le añaden los controles necesarios.

2. Definición de las propiedades de los controles u objetos que hayamos colocado en ese formulario. Estas propiedades determinan la forma estática de los controles, es decir, como son los controles y para qué sirven.

Por lo tanto se asocia código a los eventos que ocurren en estos objetos. La respuesta a estos eventos (click, doble click, una tecla pulsada, etc.) se le llama procedimiento, y debe generarse de acuerdo a las necesidades del programa.

3. Generación del código del programa. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, se ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos Procedimientos no responden a un evento asociado a un objeto, sino que responden a un evento producido durante la ejecución del programa.

1.2.8 SISTEMAS DE TIEMPO REAL.

Un sistema de tiempo real debe ser muy confiable. Su respuesta debe ser controlada, incluso, en condiciones de sobrecarga, y no puede volver atrás y reiniciar desde un contexto preexistente. Además que son de "tiempo infinito", por lo que deben poder recuperarse automáticamente de condiciones de excepción[24].

El sistema debe ejecutar las tareas que tiene asignadas, respetando las metas temporales de cada una de ellas.

Un STR no solamente es importante que las tareas se realicen, sino que también deben cumplirse en los plazos de tiempo previstos.

Típicamente un STR interactúa con su entorno, recibiendo información (eventos) del medio ambiente y realiza acciones para producir cambios en dicho entorno. En estos casos se les denomina sistemas *reactivos*, ya que reaccionan ante estímulos del medio ambiente.

Típicamente un STR no maneja una única tarea, sino varias simultáneamente, en el que se deben emplear estrategias para analizar y poder predecir el cumplimiento de las metas temporales (planificación).

Algunas de las aplicaciones son:

- Control de Procesos Industriales.
- Control de Aeronaves en vuelo.
- Sistemas de Armas.
- Control de Redes de Comunicación.
- Procesamiento de señales.

- Electrónica del automóvil.
- Robótica.
- Control de plantas nucleares.

En muchos casos son aplicaciones *criticas* (vidas humanas, dinero).

Los STR presentan la siguiente clasificación:

1) Restricciones temporales.
   - Sistemas de tiempo real duro (hard real-time)
     - Los límites de tiempo son estrictos.
     - El no cumplimiento puede tener consecuencias más o menos graves.
     - En algunos casos puede ser preferible un trabajo imperfecto pero terminado a tiempo.
     - Ejemplo: control de un reactor nuclear.
   - Sistemas de tiempo real blando (sofá real-time)
     - Los límites de tiempo son flexibles.
     - Ej.: sistema de reserva de pasajes.
   - Sistemas de tiempo real firme (firm real-time)
     - Sistemas de tiempo real duro que pueden tolerar pérdidas, si la probabilidad de ocurrencia de las mismas es baja.
2) Escalas de tiempo.
   - Basados en reloj.
     - El pasaje del tiempo.
     - Ej.: sistemas periódicos.
   - Basados en eventos.
     - Ej.: las acciones se inician a partir del cierre de una llave, o la lectura del sensor.
   - Interactivos.
     - Ej.: un operador ingresando datos.
3) Integración con el sistema físico.
   - Embebidos (o empotrados).
     - Ej.: sistema de control de inyección de combustible de un automóvil.
   - No embebidos.

- Orgánicos.
  - Independientes del hardware en que corren.
- Débilmente acoplados.
  - Pueden correr en otro hardware rescribiendo ciertos módulos.

4) Forma de procesamiento.
   - Sistemas centralizados.
     - Un único nodo (mono o multiprocesador) encargado de atender a todas las tareas.
     - Las tareas se comunican a través de memoria compartida (el gasto de tiempo en comunicación es insignificante).
   - Sistemas distribuidos.
     - Varios nodos, unidos a través de una red se reparten la atención de los distintos procesos.
     - Las tareas se comunican a través de la red, no hay memoria compartida (el gasto de tiempo en comunicación es importante).
5) Estrategia de planificación de las tareas.
   - Sistemas estáticos.
     - Todas las tareas, su naturaleza y características en tiempo de diseño se planifica la ejecución de las mismas.
     - El sistema no admite la aparición de una nueva tarea sobre la marcha.
     - Bajo costo de ejecución.
   - Sistemas dinámicos (o adoptivos).
     - Puede haber un conjunto de tareas conocido de antemano, pero ante la aparición de una nueva tarea, el sistema analiza si la puede garantizar sin afectar a las tareas que ya maneja, y en ese caso la agrega a la lista de tareas.
     - Mayor costo de ejecución, porque el planificador tiene un trabajo de análisis adicional.

BIBLIOGRAFÍA.

[1] CARBALLAR JOSÉ A. Comunicaciones del PC. ED. Alfaomega, RA-MA. 2da. Ed. México 1997, pp. 113-118.
[2] Telefonía digital. http://www.cft.gob.mx/html/la_era/intro.htm [Consulta: 4 de febrero de 2003] Formato HTM.
[3] MÉXICO DIGITAL COMUNICACIÓN. Electrónica y servicio. ED. Impresos publicitarios Mogue, Ed. México 2002. pp. 12-22
[4] Historia de la comunicación por alambres. [Consulta 5 de Enero de 2003] Formato HTM. http://www.austral.addr.com/old_wires/telegrafo_y_telefono3.htm
[5] El puerto de Impresora. [Consulta: 13 de enero de 2003] Formato HTM. http://www.ii.uam.es/~gdrivera/varios/notas_lpt.htm
[6] CREATUROIDES ROBOTICA FANTÁSTICA. El puerto paralelo de la PC [consulta: 13 de enero de 2003] Formato HTM. http://www.creaturoides.com/paralelo.htm
[7] Tecnología del PC. [Consulta 15 enero de 2003]. Formato HTM. http://www.zator.com/Hardware/H2_5_3.htm
[8] Bus. [Consulta 16 enero de 2003]. Formato HTM. http://www.geocities.com/nubebosa/actual/bus1.htm
[9] Conjunto de comandos [Consulta 17 enero de 2003]. Formato HTM. http://support.ap.dell.com/docs/comm/47cjx/sp/command.htm
[10] Programación de los registros [Consulta 19 enero 2003]. Formato HTML. http://www.stalker.es/personal/tcpip/modemweb/modem/registross.html
[11] MODEMS. [Consulta 20 enero 2003]. Formato HTML. http://www.sc.ehu.es/acwbecae/ikasgaiak/CDA/Lab3.html
[12] Codificación y decodificación DTMF. [Consulta 22 enero 2003]. Formato HTM. http://www.plazacolima.com/tecnoplaza/rformas/Articulo01/pagina4.htm
[13] Receptor Autónomo DTMF por Puerto Paralelo. [Consulta 25 enero 2003]. Formato HTM. http://www.pablin.com.ar/electron/circuito/computer/dtmfrx/index.htm
[14] M-8870-01. [Consulta 7 Febrero 2003]. Formato HTM. www.clare.com
[15] Introducción al correo electrónico. [Consulta 26 enero 2003]. Formato HTM. http://tejo.usal.es/~nines/d.alumnos/correo6/intro.htm
[16] Elección del protocolo a emplear (POP vs IMAP). [Consulta 28 enero 2003]. Formato HTML. http://www.ua.es/es/servicios/alumnos/email/config/popimap.html
[17]SMTP. [Consulta 28 enero 2003]. Formato HTML. http://ditec.um.es/laso/docs/tut-tcpip/3376c46.html
[18]SMTP utilizando sockets. [Consulta 28 enero 2003]. Formato HTM. http://www.programacion.com/php/articulo/fli_phpsmtp/
[19]Comandos SMTP. [Consulta 28 enero 2003]. Formato HTML. http://penta.ufrgs.br/rc952/trab1/smtpcomd.html
[20]Protocolos. [Consulta 28 enero 2003]. Formato HTML. http://burete.forodigital.es/angel/programacion/Redes/Protocolos/Pop3.html
[21]http://www.fortunecity.es/arcoiris/tarot/241/rfc/rfc1082.htm
[22] Post Office Protocol - Version 3. [Consulta 28 enero 2003]. Formato TXT. http://ietf.org/rfc/rfc1521.txt

[23] CEVALLOS SIERRA FCO. JAVIER. Curso de programación de Visual Basic 6. ED. Alfaomega, RA-MA. 1da. Ed. México 2000, pp. 1-84.
[24]Software de tiempo real. [Consulta 9 febrero 2003]. Formato HTM. 2003http://www.frbb.utn.edu.ar/electronica/treal/

CAPÍTULO II

DESARROLLO DE HARDWARE

2.1 INTRODUCCIÓN.

La incorporación de dispositivos periféricos proporcionan una gran variedad de características y funciones que permiten hacer una herramienta más cómoda y eficaz. Entre otras cosas se utiliza la línea telefónica de forma automática para establecer comunicación con el exterior, así como el puerto paralelo como interfaz para la detección en la pulsación de tonos telefónicos y así procesar ordenes remotas. Asimismo la capacidad de detección de señales provenientes de diversos dispositivos periféricos garantiza que la información intercambiada entre el exterior y un sistema digital sea rápido y eficaz.

Por último, el almacenamiento de información en un dispositivo de memoria secundaria se consigue mediante una codificación especial con la finalidad de disminuir la cantidad de bits de información, con lo que en la practica supone un aumento de la velocidad de transmisión de información.

2.2 REQUERIMIENTOS DE HARDWARE.

Los componentes básicos requeridos para el desarrollo y buen funcionamiento del sistema esta dado por los siguientes elementos:

- Procesador Pentium II a 350 Mh.
- 64 Mb en RAM. Sin embargo se recomienda tener 256 Mb.
- 20 Mb de espacio libre en disco duro.
- 2 puertos paralelos.
- 2 puertos USB.
- Línea telefónica.
- MODEM compatible con Windows y con capacidad de identificador de llamadas (Contrato con la compañía telefónica del servicio identificador de llamadas).
- Servicio de Internet.
- Tarjeta interfaz DTMF para la decodificación de tonos.
- Láser clase II.
- 6 Reflectores (Dependiendo perímetro del área a proteger).

En los siguientes puntos se presenta el desarrollo de cada uno de los componentes que intervienen en la investigación realizada, tal y como se muestra la jerarquía de funciones en la figura 2.2.1.

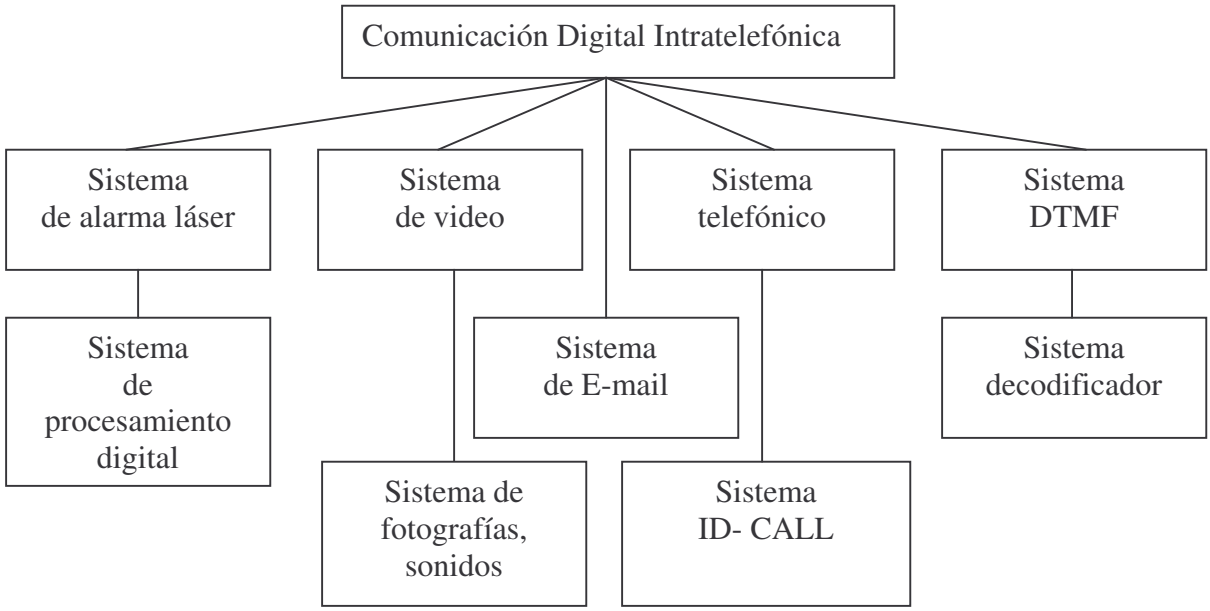


Figura 2.2.1 Esquema jerárquico del sistema.

El sistema de alarma es el encargado de prender o apagar el láser que es el medio de seguridad en una casa, el perímetro a rodear puede ser de hasta 300 metros. A la recepción de luz se le aplica procesamiento digital para determinar el momento en que ocurre la interrupción de luz y activar la alarma local o remota. También se cuenta el sistema de video cuya función es monitorear la puerta principal de una casa y fotografiar cada vez que se toque el timbre, guardando las fotografías para su posterior consulta.

El sistema telefónico es la contestadora de llamadas entrantes que con apoyo de una base de datos y un identificador de llamadas las clasifica en tres tipos:

- Llamadas aceptadas. Las cuales son contestadas al segundo Ring.
- Llamadas rechazadas. Las cuales son cortadas inmediatamente al segundo Ring.
- Llamadas normales. Las cuales no están en la base de datos, pero son contestadas al 5to. Ring.

El sistema de E-Mail es el encargado de enviar automáticamente las llamadas guardadas así como las fotografías tomadas para mantener a los integrantes de la casa informados aún en su ausencia.

Se cuenta con un sistema DTMF (Multi Frecuencia de Dos Tonos) cuya función es recibir los pulsos del teclado de un teléfono digital decodificarlo y realizar la tarea previamente indicada.

El DTMF también se utiliza para recibir mensajes cifrados cuyos sonidos son decodificados y almacenados como texto.

2.3 PUERTO PARALELO.

El puerto paralelo[1] es utilizado como un medio de entrada/salida de datos que interactúa con dos interfaces mostradas más adelante. El puerto cuenta con tres tipos de direcciones cada una de ellas se caracteriza por realizar una tarea en específico.

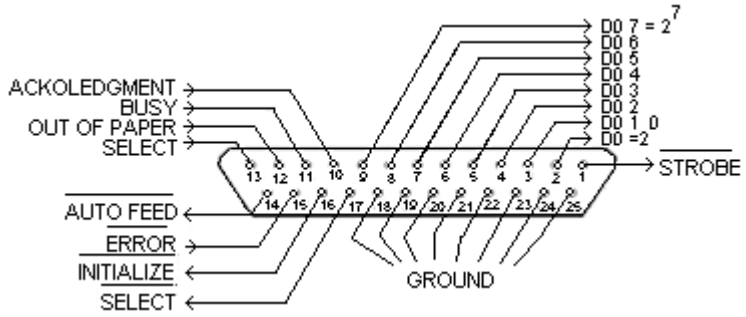


Figura 2.3.1 Puerto paralelo DB25

El puerto paralelo cuenta con tres tipos de direcciones cada una de ellas se describen a continuación:

Dirección 378H: En esta dirección la CPU escribe los *datos* que serán enviados. Es un puerto de salida[2], los pines utilizados se muestran en la tabla 2.3.1.

NÚMERO DE PIN	SEÑAL
2	DO 0
3	DO 1
4	DO 2
5	DO 3
6	DO 4
7	DO 5
8	DO 6
9	DO 7

Tabla 2.3.1 Puerto de datos.

Dirección 379H: Este es un puerto de *entrada*. A través de este puerto la CPU puede conocer el *estado* en que se encuentra el dispositivo periférico, los pines utilizados se muestran en la tabla 2.3.2.

NÚMERO DE PIN	SEÑAL
10	ACKNOLEDGMENT
11	BUSY
12	OUT OF PAPER
13	SELECT
15	ERROR

Tabla 2.3.2 Puerto de entrada.

Dirección 37AH: En esta dirección el ordenador escribe las señales que *controlan* la información de salida, los pines utilizados son 1,14,16 y 17. Sin embargo en este proyecto no se requirió de esta dirección de puerto.

2.4 ALARMA LÁSER.

El sistema de alarma es controlado localmente o remotamente por una computadora, la propuesta planteada es conectar al puerto paralelo DB25 un láser Clase II. La configuración básica utilizada se muestra en la figura 2.4.1.

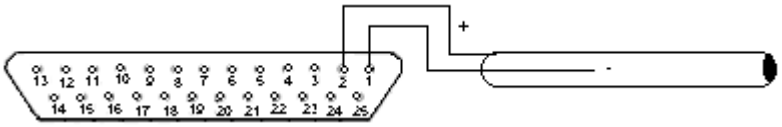


Figura 2.4.1 Conexión de láser al DB25

Sin embargo para aumentar la intensidad y alcance de la luz láser se complementa con una pila de 9v. y un relevador tal y como se muestra en la siguiente figura.

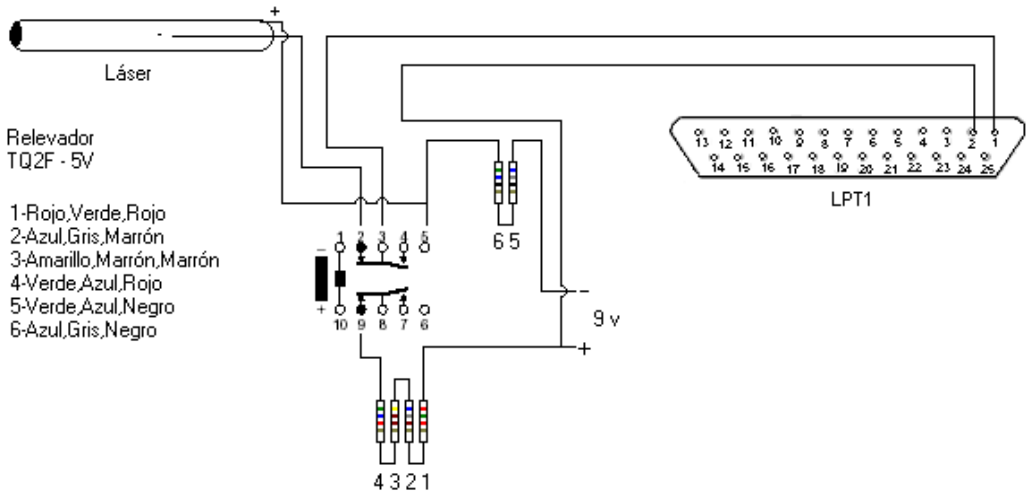
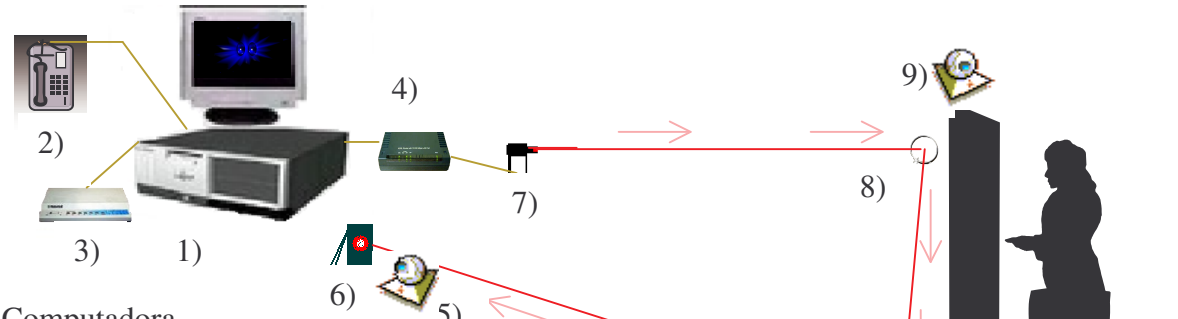


Figura 2.4.2 Conexión puerto paralelo, relevador, láser.

Mediante reflexión se rodea el perímetro que se desea asegurar, convirtiendo parte del sistema en un vigilante silencioso de todo un hogar.

Una vez cubierto el perímetro como se muestra en la figura 2.4.3, el haz de luz generado por el láser debe llegar al lugar donde será monitoreado por una cámara web, el haz de luz no debe llegar de forma directa a la cámara ya que puede llegar a dañarla, por lo tanto se recomienda proyectarla como punto final en una pared u objeto no reflexivo.



1) Computadora.
2) Línea telefónica.
3) Módem con identificador de llamadas.
4) Decodificador de tonos.
5) Cámara web para monitoreo del láser.
6) Paleta para contener luz láser.
7) Rayo láser clase II.
8) Espejos.
9) Cámara para fotografiar personas en la puerta principal.

Figura 2.4.3 a) Diagrama a bloques de protección mediante láser.

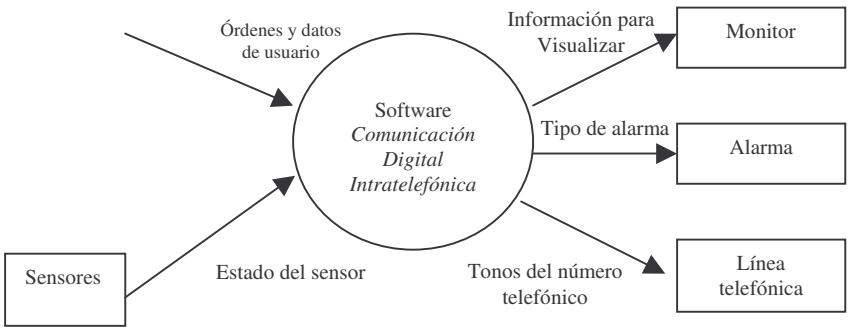


Figura 2.4.3 b) Diagrama de nivel contextual.

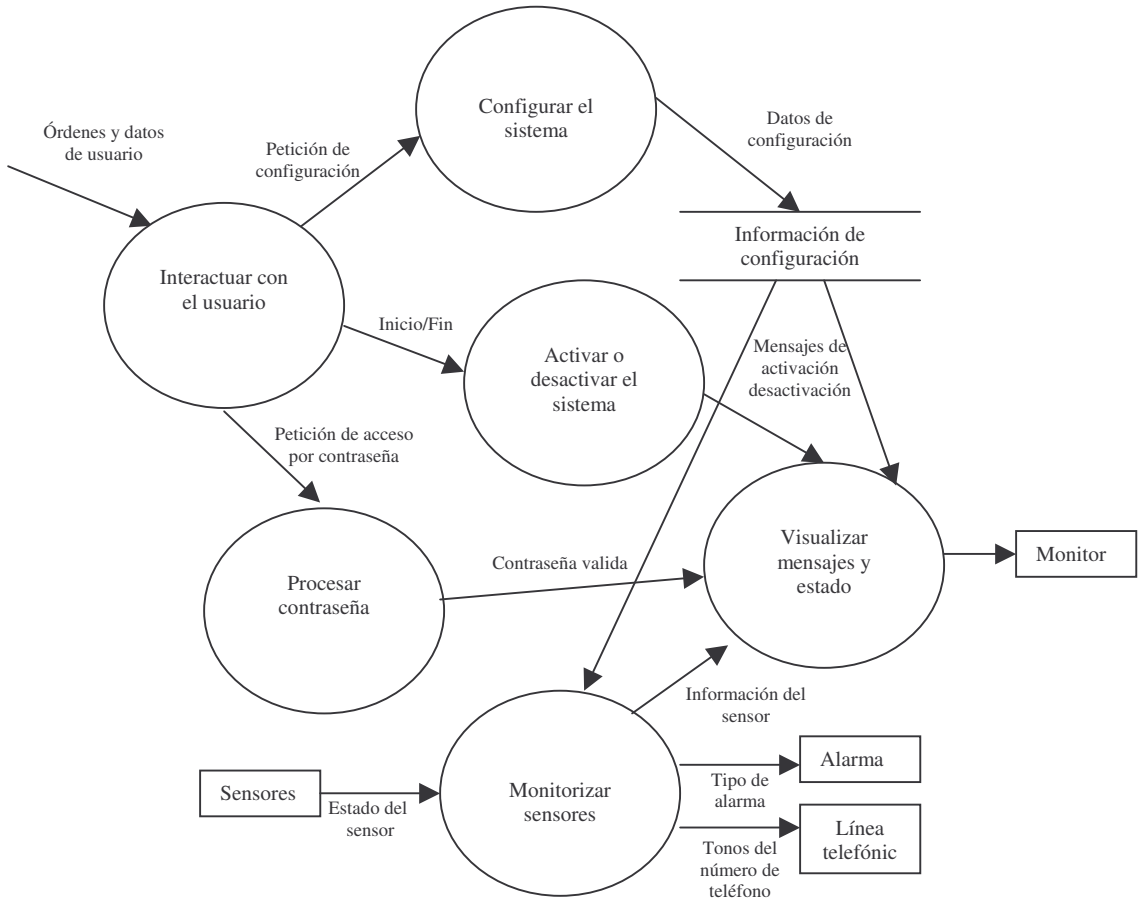


Figura 2.4.3 c) Diagrama de nivel 1.

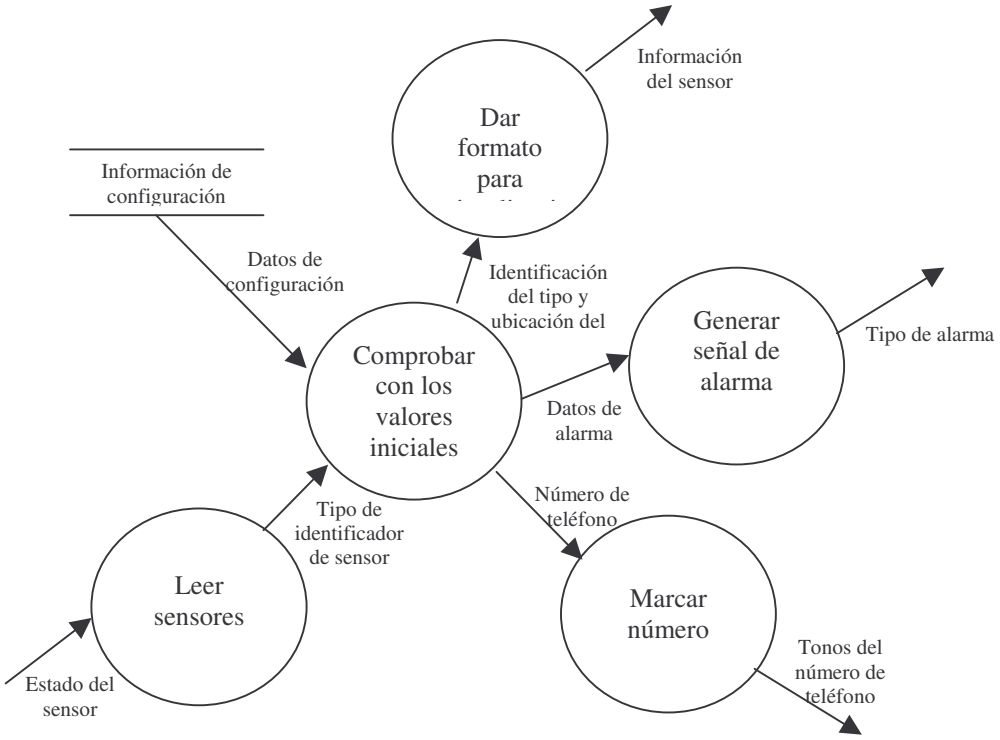


Figura 2.4.3 d) Diagrama de nivel 2 para monitorizar sensores.

La figura 2.4.4 muestra la luz captada por la cámara web proyectada en una pared, se observa que alrededor se percibe un color rojizo sin embargo el centro es captado como un blanco intenso proporcionando así el área a monitorear.

Figura 2.4.4 Láser captado por cámara web

2.5 DECODIFICADOR DTMF (MULTI FRECUENCIA DE DOS TONOS).

La elaboración de un circuito detecta el tono generado por la pulsación de un digito del teclado telefónico, tiene como finalidad que una computadora realice tareas ordenadas remotamente mediante un teléfono digital.

La detección de los tonos se realiza empleando una interfaz con el circuito integrado M-8870-01 y el puerto paralelo de la computadora, el circuito mostrado en la figura 2.5.1 se encarga de decodificar 16 tonos telefónicos.

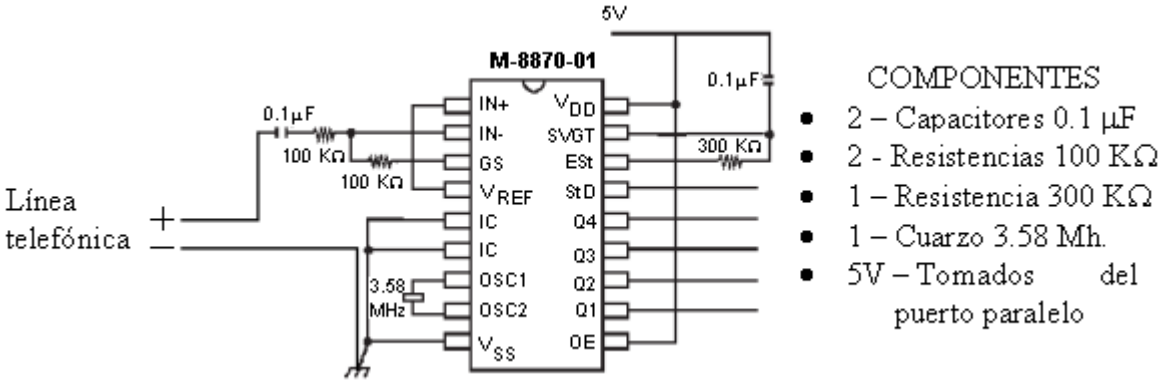


Figura 2.5.1 Decodificador DTMF.

La pulsación de uno de los dígitos del teclado telefónico da como resultado la generación de dos frecuencias clasificadas en: frecuencia baja y frecuencia alta.

Dichas frecuencias son decodificadas por el circuito M-8870-01 devolviendo un valor en formato binario[3]. La tabla 2.5.1 muestra cada uno de los valores devueltos de acuerdo a las frecuencias percibidas por el circuito integrado.

FBAJA	FALTA	TECLA	Q4	Q3	Q2	Q1
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1477	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1477	8	1	0	0	0
852	1209	9	1	0	0	1
941	1336	0	1	0	1	0
941	1209	S	1	0	1	1
941	1477	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

Tabla 2.5.1 Valores del circuito M-8870-01

La figura 2.5.2 muestra el circuito básico para detectar un tono telefónico, eliminando ruido mediante un transformador de 600 ohms.

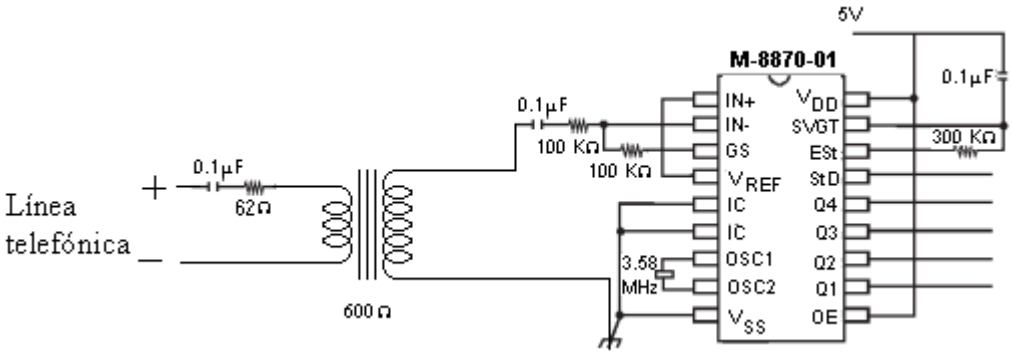


Figura 2.5.2 Decodificador DTMF sin ruido.

El circuito M-8870-01 cada vez que recibe una frecuencia analógica valida de acuerdo a los rangos presentados en la tabla 2.5.1 genera una salida digital en forma binaria representado por Q1, Q2, Q3, y Q4, además de utilizar StD como indicador para la computadora que se ha producido un nuevo valor.

Al interactuar el decodificador de tonos con el LPTn, se utiliza la configuración mostrada en la figura 2.5.3. De tal forma que la computadora lee el valor generado por el DTMF, realizando la acción correspondiente.

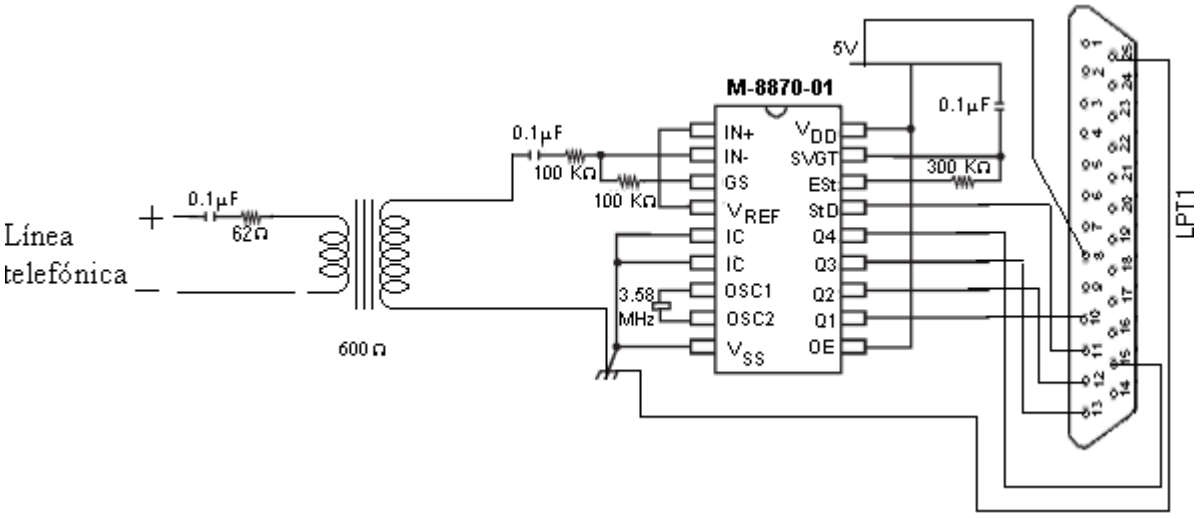


Figura 2.5.3 Configuración para deposito de valores en LPT1.

2.6 ENTRADA/SALIDA DE SONIDO PC-TELÉFONO/TELÉFONO-PC.

La interfaz de acoplamiento de sonido entre la computadora-teléfono se logra con éxito mediante la implementación de los componentes mostrados en la figura 2.6.1. El transformador de 600 Ω[4] tiene la funcionalidad de equilibrar la impedancia entre el puerto de sonido y la línea telefónica.

La implementación de acoplamiento con el teléfono es estrictamente necesario, ya que si no se emplea el transformador se ocasiona un daño a la tarjeta de sonido y bocinas.

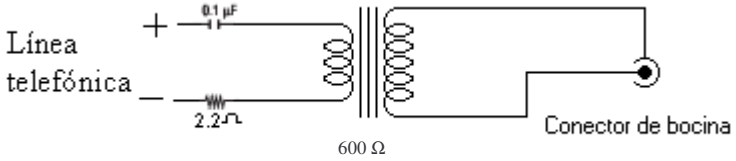


Figura 2.6.1 Interfaz de acoplamiento Computadora-Teléfono.

El acoplamiento telefónico se realiza con la finalidad de recibir y emitir sonidos o voz en forma directa, de la computadora al teléfono o viceversa, para su procesamiento correspondiente.

La computadora se comporta como un receptor cuando se permite la grabación de mensajes provenientes del teléfono o micrófono local de la PC. Y como emisora al momento que entra una llamada telefónica, y es contestada emitiendo un mensaje o sonido de bienvenida.

BIBLIOGRAFÍA.

[1] Puerto paralelo de la PC. [Consulta 9 febrero 2003]. Formato HTM.
http://www.austral.edu.ar/web/ingenieria/LCD2002/Info/PuertoParalelo.htm
[2] Señales empleadas por el puerto paralelo. [Consulta 9 febrero 2003]. Formato HTM.
http://www.terra.es/tecnologia/articulo/html/tec4788.htm
[3] Decodificación binaria de tonos. [Consulta 9 febrero 2003]. Formato HTML.
http://www.compdist.com/Manufacturers/teltone/dtmfrec.html
[4] Interconexión con la tarjeta de sonido de la computadora. [Consulta 11 febrero 2003]. Formato HTM. http://pages.cthome.net/n1mm/html/Spanish/Interfacing_EA.htm

CAPÍTULO
III

DESARROLLO DE SOFTWARE

3.1 INTRODUCCIÓN.

Uno de los lenguajes más sencillos, utilizados y difundidos desde la aparición del PC ha sido el lenguaje Basic en sus distintas versiones, es por ello que el sistema se genera en una versión más potente como es Visual Basic 6.0.

Una de las características principales por lo que se selecciono VB es por las condiciones que ofrece para interactuar con las principales librerías dll (biblioteca de vínculo dinámico) que componen al sistema operativo Windows, así como la facilidad para el manejo de los puertos de comunicaciones principales mediante comandos lo suficientemente potentes como para afrontar tareas especificas con el exterior.

Así, el lenguaje Basic permite al programador realizar programas con un alto grado de abstracción, manteniendo un control significativo sobre el funcionamiento del ordenador.

3.2 REQUERIMIENTOS DE SOFTWARE.

Los componentes básicos requeridos para el desarrollo y buen funcionamiento del sistema, esta dado por el siguiente software:

- Sistema operativo Windows 98/Windows NT/Windows 2000/Windows XP.
- Visual Basic 6.0 Profesional/Empresarial.
- Drivers de las cámaras web utilizadas.

3.3 PUERTO PARALELO.

El lenguaje de programación utilizado es Visual Basic, una de las características del lenguaje es que al realizar una función en particular no existente en VB, ésta debe implementarse como una librería DLL[1].

Tal es el caso de la existencia de una función en la que se requiere interactuar con el puerto paralelo, al no existir dicha función en el lenguaje, es necesario implementarlo creando un archivo de biblioteca de vínculo dinámico (DLL, dynamic linked library) que permita leer y escribir en el puerto en sus direcciones correspondientes.

La librería es creada en Visual C++ 6. 0 el punto de entrada y de salida en una DLL en Win32 es una función denominada *DllMain*; ésta es una de las diferencias con respecto a Win16. La función es opcional; esto quiere decir que si no se escribe, el compilador asume que no hace nada, simplemente retorna *TRUE*.

DllMain utiliza el convenio de llamada *WinApi* para sus tres parámetros en lugar de *FAR PASCAL* que ha quedado obsoleto. El prototipo de esta función es el siguiente:

BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)

La función retorna el valor *TRUE* para indicar que se ha ejecutado satisfactoriamente. Si durante el proceso de iniciación la función retorna *FALSE* el sistema cancela el proceso.

El parámetro *fdwReason* indica la razón por la que ha sido llamada la función *DllMain*: iniciación o terminación, por un proceso o por un hilo (thread).

La siguiente tabla describe el significado de los posibles valores del parámetro fdwReason.

Valor de fdwReason	Descripción
DLL_PROCESS_ATTACH	Un nuevo proceso intenta acceder a la DLL; se asume un hilo.
DLL_THREAD_ATTACH	Un nuevo hilo de un proceso existente intenta acceder a la DLL; esta llamada se hace a partir del segundo hilo de un proceso vinculado a la DLL.
DLL_PROCESS_DETACH	Un proceso abandona la DLL.
DLL_THREAD_DETACH	Uno de los hilos adicionales (no el primer hilo) de un proceso abandona la DLL.

Tabla 3.3.1 Valores del parámetro fdwReason.

El parámetro pvReserved se reserva para ser utilizado por el sistema.

El parámetro hInstance es el *handle* de la DLL.

Los pasos para crear la DLL se describe a continuación:

3.3.1 ARCHIVO DE CABECERA (.h).

Los ficheros de cabecera contienen declaraciones y definiciones que el preprocesador de C incluye en el fichero fuente justo antes de la compilación. Para el proyecto se escribe un fichero *puerto.h* que contiene las declaraciones de las funciones *Escritura* y *Lectura*.

```
//*************************************************************************
//*Nombre del archivo: Puerto.h                                           *
//*                                                                       *
//*Este archivo de cabecera contiene las funciones prototipo para las funciones exportables  *
//*por la DLL.                                                            *
//*************************************************************************
#define EXPORT extern "C" __declspec(dllexport)
//*************************************************************************
//*  Escritura(int direccion_puerto_LPT1, int valor)                      *
//*************************************************************************
EXPORT BOOL Escritura(int, int);
//*************************************************************************
// * Lectura(int direccion_puerto_LPT1)                                   *
//*************************************************************************
EXPORT int Lectura(int);
```

3.3.2 ARCHIVO FUENTE (.c o .cpp).

Fundamentalmente, el fichero fuente contiene la definición de las funciones. El siguiente archivo *puerto.cpp* contiene los archivos de cabecera *windows.h* y *dos.h*, la función de entrada y de salida de la DLL, *DllMain*, y las funciones que se han incluido en la biblioteca *puerto.h*, E*scritura* y *Lectura*.

```
//*************************************************************************
//*  Nombre del archivo: Puerto.cpp                                       *
//* DLL para manejo de puerto paralelo                                    *
//*************************************************************************
#include <windows.h>
#include <dos.h>
#include "puerto.h"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
   return TRUE;
}
```

```
EXPORT BOOL Escritura(int direccion, int valor)
{
   int test;
   test = outp(direccion, valor);
   if(test != valor)  return FALSE;
   return TRUE;
}
EXPORT int Lectura(int direccion)
{
   int numero;
   numero = inp(direccion);
   return numero;
}
```

3.3.3 ARCHIVO DE DEFINICIÓN DE MÓDULOS (.def).

El archivo de definición de módulos informa al enlazador (linker) sobre cómo crear el archivo ejecutable. A continuación se muestra el código del módulo.

```
//*************************************************************************
//*Nombre del archivo: Puerto.def                                         *
//*************************************************************************
LIBRARY          "Puerto"
DESCRIPTION      'Puerto'
EXETYPE          NT
SUBSYSTEMWINDOWS
STUB          'WINSTUB.EXE'
VERSION             1.0
CODE          EXECUTE READ
DATA          READ WRITE
HEAPSIZE    1048576,4096
EXPORTS
        LECTURA=Lectura @1
        ESCRITURA=Escritura @2
```

3.3.4 LLAMADA DE LA FUNCIÓN DLL DESDE VISUAL BASIC.

Una vez creada la librería en Visual C++, para acceder a la DLL[2] desde Visual Basic se escribe el siguiente código:

```
Rem En el módulo GENERAL:
Option Explicit
Private Declare Function Escritura Lib   "C:\Puerto\Puerto.dll"   (ByVal direccion As Long, _
ByVal Valor As Long) As Bolean
Private Declare Function Lectura Lib  "C:\Puerto\Puerto.dll"   (ByVal dir2 As Long) As Long
Private sub Escritura_Puerto( )
  Dim Result as Boolean
  Result = Escritura(Direccion_Puerto, Valor_Enviado)
End Sub
Private sub Lectura_Puerto( )
  Dim Valor as Integer
  Valor= Lectura(Direccion_Puerto)
End Sub
```

3.4 ACTIVAR CÁMARA WEB.

Para enlazar la cámara web con la aplicación se emplea la librería avicap32.DLL[3,4] que forma parte del sistema operativo y reutilizada a través de Visual Basic con el código que se describe en las siguientes líneas.

```
'//OBLIGAR QUE TODAS LAS VARIABLES UTILIZADAS SEAN DECLARADAS
'//PREVIAMENTE
Option Explicit
'//Evento que se ejecuta antes de presentar el formulario
Private Sub Form_Load()
'//Declaración de variables locales del procedimiento
Dim Nombre As String * 100
Dim Version As String * 100
```

```
    '//CREAR CAPTURA EN VENTANA
    capGetDriverDescriptionA 0, Nombre, 100, Version, 100  'Información del driver
    Identificador = capCreateCaptureWindowA(Nombre, TipoVentana Or FramesContinuos Or _
    VentanaVisible Or VentanaHilo, 0, 0, 160, 120, Me.hwnd, 0)
    '//ASIGNAR TITUO DE VENTANA CON EL NOMBRE DEL DRIVER
    SetWindowText Identificador, Nombre
    '//ASIGNAR EL FLUJO DE VIDEO CON LA FUNCION MANTENER LLAMADA
    LlamadaCapturaEstadoOn Identificador, AddressOf MiEstadodeLlamada _
    AsiganErrorLlamada Identificador, AddressOf MiEstadodeError
    If DriverConectado(Identificador, 0) Then
       '//CAMBIA LA ESCALA EN ON
       EscalaPrevia Identificador, True
       '//ASIGNA UN TIEMPO EN MILISEGUNDOS ANTES DE PRESENTAR LA IMAGEN
       FramesPrevioCaptura Identificador, 66
       '//INICIA  LA PREPARACIÓN DE LA IMAGEN ENVIADA POR LA CAMARA
       CapturaPrevia Identificador, True
       '//REDIMENSIONA Resize LA VENTANA DE CAPTURA PARA MOSTRAR LA IMAGEN
       RedimensionaVentanaCaptura Identificador
    End If
End Sub
'//EL SIGUIENTE CÓDIGO SE DEBE ESCRIBIR EN UN MODULO (.bas)
        Type ApuntadorApi
            x As Long
            y As Long
        End Type
        Public Identificador As Long          '//MANIPULACION DE LA VENTANA DE CAPTURA
        '//DEFINE EL INICIO DE RANGO DEL MENSAJE
        Public Const Inicio_Direccion = &H400
        Public Const MantenerLlamada_Estado = Inicio_Direccion + 3
        Public Const MantenerLlamada_Error = Inicio_Direccion + 2
        Public Const DireccionDel_Driver = Inicio_Direccion + 10
        Public Const LeeDriverDeCaptura = Inicio_Direccion + 14
        Public Const AsignaEscala = Inicio_Direccion + 53
```

```
Public Const AsignaCantidadPreviaFrames = Inicio_Direccion + 52
Public Const AsignaCapturaPrevia = Inicio_Direccion + 50
Public Const CXFrame = 32
Public Const CYFrame = 33
Public Const HWND_Abajo = 1
Public Const NoMover = &H2
Public Const NoZorder = &H4
Public Const LeeEstado = Inicio_Direccion + 54
Public Const TipoVentana = &HD00000
Public Const FramesContinuos = &H40000
Public Const VentanaVisible = &H10000000
Public Const VentanaHilo = &H40000000
Type CapturaEstado
   AnchoImagen As Long                 '// ANCHO DE LA IMAGEN
   AlturaImagen As Long                '// ALTURA DE LA IMAGEN
   MostrarVentana As Long              '// PRESENTACION PREVIA DEL VIDEO
   VentanaOp As Long                   '// OPACAR EL VIDEO
   fEscala As Long                     '// ESCALA DE LA IMAGEN
   Desplazamiento As ApuntadorApi      '// POSICION DE DESPLAZAMIENTO
   UsarPaletaDefault As Long           '// USAR POR DEFAULT LA PALETA DEL DRIVER
   AudioHardware As Long               '// PRESENTACION DE AUDIO POR HARDWARE
   ArchivoExiste As Long               '// SOBREESCRIBIR ARCHIVO DE VIDEO SI EXISTE
   FramesVideoAdmitidos As Long        '// FRAMES DE VIDEO CAPTADOS
   FramesVideoAdmitidossPendientes As Long     '// FRAMES DE VIDEO PENDIENTES
   WavAdmitido As Long                 '// wave CAPTADO
   TiempoAdmitido As Long          '// DURACION DE CAPTURA DURANTE EL ENLACE
   PaletaAdmitida As Long              '// COLORES ADMITIDOS DURANTE EL USO
   CapturaAhora As Long                '// CAPTURA EN PROGRESO
   RetornoError As Long                '// ERROR DURANTE LA OPERACION
   NumeroVideoBuffer As Long           '// NUMERO ACTUAL DE VIDEO EN LA MEMORIA
                                       '// INTERMEDIA
   NumeroAudioBuffer As Long           '// NUMERO ACTUAL DE AUDIO EN LA MEMORIA
                                       '// INTERMEDIA
End Type
```

```
'// DOS FUNCIONES EXPORTADAS POR AVICap
Declare Function capCreateCaptureWindowA Lib "avicap32.dll" ( _
   ByVal lpszWindowName As String, _
   ByVal dwStyle As Long, _
   ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Integer, _
   ByVal hWndParent As Long, ByVal nID As Long) As Long
Declare Function capGetDriverDescriptionA Lib "avicap32.dll" ( _
   ByVal wDriver As Integer, _
   ByVal Nombre As String, _
   ByVal cbName As Long, _
   ByVal Version As String, _
   ByVal cbVer As Long) As Boolean
'//MANIPULAION DE LA VENTANA
Declare Function SetWindowText Lib "user32" Alias "SetWindowTextA" (ByVal hwnd As Long,
ByVal lpString As String) As Long
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long,
ByVal wMsg As Long, ByVal wParam As Integer, ByVal lParam As Long) As Long
Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal hWndInsertAfter
As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal
wFlags As Long) As Long
'//MANIPULACION DE LA MEMORIA
Declare Function lStrCpy Lib "kernel32" Alias "lstrcpyA" (ByVal lpString1 As Long, ByVal
lpString2 As Long) As Long
Function MiEstadodeLlamada(ByVal lwnd As Long, ByVal iID As Long, ByVal ipstrStatusText
As Long) As Long
Dim EstadoTexto As String
Dim uEstadoTexto As String
  If iID = 0 Then Exit Function
  '//CONVIERTE EL APUNTADOR EN MODO REAL A CADENA
  EstadoTexto = String$(255, 0)                    '// HACE EL ALOJAMIENTO DEL MENSAJE
  lStrCpy StrPtr(EstadoTexto), ipstrStatusText   '// COPIA EL MENSAJE DENTRO DE LA
                                                   '// CADENA
  EstadoTexto = Left$(EstadoTexto, InStr(EstadoTexto, Chr$(0)) - 1)  '// LA CERRADURA
```

```
                                                    '// SOLO ES NULA POR LA IZQUIERDA
  uEstadoTexto = StrConv(EstadoTexto, vbUnicode)          '// CONVIERTE A MODO
                                                    '//UNICODE
End Function
Function MiEstadodeError(ByVal lwnd As Long, ByVal iID As Long, ByVal ipstrStatusText As
Long) As Long
Dim EstadoTexto As String
Dim uEstadoTexto As String
  If iID = 0 Then Exit Function
  '//CONVIERTE EL APUNTADOR EN MODO REAL A CADENA
  EstadoTexto = String$(255, 0)
  lStrCpy StrPtr(EstadoTexto), ipstrStatusText
  EstadoTexto = Left$(EstadoTexto, InStr(EstadoTexto, Chr$(0)) - 1)
  uEstadoTexto = StrConv(EstadoTexto, vbUnicode)
  LogError uEstadoTexto, iID
End Function
Sub RedimensionaVentanaCaptura(ByVal lwnd As Long)
Dim CapturaEstado As CapturaEstado
Dim lCaptionAlto As Long
Dim lX_Borde As Long
Dim lY_Borde As Long
    lCaptionAlto = GetSystemMetrics(4)
    lX_Borde = GetSystemMetrics(CXFrame)
    lY_Borde = GetSystemMetrics(CYFrame)
    '//ASIGNA ATRIBUTOS A LA VENTANA DE CAPTURA COMO EL ANCHO, LA
    '//ALTURA
    If EstadoCaptura(lwnd, VarPtr(CapturaEstado), Len(CapturaEstado)) Then
      '//REDIMENSIONA EL TAMAÑO DE LA VENTANA DE CAPTURA
      SetWindowPos lwnd, HWND_Abajo, 0, 0, _
        CapturaEstado.AnchoImagen + (lX_Borde * 2), _
        CapturaEstado.AlturaImagen + lCaptionAlto + (lY_Borde * 2), _
        NoMover Or NoZorder
    End If
End Sub
```

```
Function LlamadaCapturaEstadoOn(ByVal lwnd As Long, ByVal lpProc As Long) As Boolean
  LlamadaCapturaEstadoOn = SendMessage(lwnd, MantenerLlamada_Estado, 0, lpProc)
End Function
Function AsiganErrorLlamada(ByVal lwnd As Long, ByVal lpProc As Long) As Boolean
  AsiganErrorLlamada = SendMessage(lwnd, MantenerLlamada_Error, 0, lpProc)
End Function
Function DriverConectado(ByVal lwnd As Long, ByVal i As Integer) As Boolean
  DriverConectado = SendMessage(lwnd, DireccionDel_Driver, i, 0)
End Function
Function EscalaPrevia(ByVal lwnd As Long, ByVal f As Boolean) As Boolean
  EscalaPrevia = SendMessage(lwnd, AsignaEscala, f, 0)
End Function
Function CapturaPrevia(ByVal lwnd As Long, ByVal f As Boolean) As Boolean
  CapturaPrevia = SendMessage(lwnd, AsignaCapturaPrevia, f, 0)
End Function
Function FramesPrevioCaptura(ByVal lwnd As Long, ByVal wMS As Integer) As Boolean
  FramesPrevioCaptura = SendMessage(lwnd, AsignaCantidadPreviaFrames, wMS, 0)
End Function
Function EstadoCaptura(ByVal lwnd As Long, ByVal s As Long, ByVal wSize As Integer) As
Boolean
    EstadoCaptura = SendMessage(lwnd, LeeEstado, wSize, s)
End Function
```

A continuación se muestra otra forma de recibir las imágenes de la cámara web en el objeto formulario.

```
'//EL SIGUIENTE CÓDIGO SE DEBE ESCRIBIR EN UN MODULO (.bas)
Public Declare Function SendMessage Lib "USER32" Alias "SendMessageA" (ByVal hwnd As
Long, ByVal Mensaje As Long, ByVal Param1 As Long, Param2 As Any) As Long
Public Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias
"capCreateCaptureWindowA" (ByVal Nombre As String, ByVal Estilo As Long, ByVal X As
Long, ByVal Y As Long, ByVal Ancho As Long, ByVal Alto As Long, ByVal Parametro As
Long, ByVal nID As Long) As Long
```

```
Public Captar As Long
Public Const CONECTAR As Long = 1034
Public Const DESCONECTAR As Long = 1035
Public Const ASIGNA_FRAME As Long = 1084
Public Const COPIA As Long = 1054
Private Sub Command1_Click()
  Command1.Enabled = False
  Command2.Enabled = True
  '//CONFIGURAR LA CAPTURE EN LA VENTANA
  Captar = capCreateCaptureWindow("CapturaWeb", 0, 0, 0, 320, 240, Me.hwnd, 0)
  '//CONECTAR AL DISPOSITIVO DE CAPTURA
  DoEvents: SendMessage Captar, CONECTAR, 0, 0
  Timer1.Enabled = True
End Sub
Private Sub Command2_Click()
  Command1.Enabled = True
  Command2.Enabled = False
  Timer1.Enabled = False
  '//DESCONECTA LA CAMARA
  DoEvents: SendMessage Captar, DESCONECTAR, 0, 0
End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
  If Command2.Enabled = False Then
    DoEvents: SendMessage Captar, DESCONECTAR, 0, 0
  End If
End Sub

Private Sub Timer1_Timer()
  On Error Resume Next
  '//LEE EL FRAME DE LA CAMARA
  SendMessage Captar, ASIGNA_FRAME, 0, 0
```

```
  '//COPIA EL FRAME AL PORTAPAPELES
  SendMessage Captar, COPIA, 0, 0
  '//ASIGNA LO QUE TENGA EL PORTAPAPELES AL OBJETO FORMULARIO
  Form1.Picture = Clipboard.GetData
  'LIMPIA EL PORTAPAPELES
  Clipboard.Clear
End Sub
```

3.5 ALARMA LÁSER.

Una vez que la cámara transmite video y éste es mostrado en algún objeto de la aplicación tal y como se muestra en la figura 3.5.1.

Figura 3.5.1 Láser captado por cámara web

El siguiente paso es calcular el centro de la imagen tal como en la figura 3.5.6, para ello se aplica un barrido con la finalidad de identificar la intensidad adecuada de luz del rayo láser. El valor de intensidad valido reconocido como luz del rayo láser es el 16777215, ya que al leer el color mediante la instrucción point devuelve un valor de 0 si el color es negro y 16777215 que es el valor máximo cuando el color es blanco.

Para encontrar y calcular el centro de la luz se aplica el barrido en un área de búsqueda de 3382 píxeles para ello se emplean 4 ciclos de la siguiente forma[5,6]:

El primer ciclo se encarga de encontrar la intensidad valida en la parte superior, como se muestra en la figura 3.5.2.

```
For ejey = 139 To 228 Step 2
  For ejex = 195 To 233 Step 2
    Valor = Picture1.Point(ejex, ejey)
    Valtot = Valor
    If Valor <> -1 Then
      If Valor = 16777215 Then
       Porcen = Valtot * (Val(Combo1.Text) / 100)
       If k = 0 Then
         superior = ejex
         Exit For
       End If
      End If
    End If
  Next ejex
  DoEvents
```

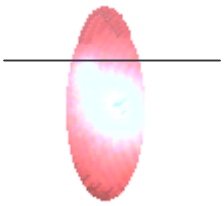


Figura 3.5.2 Barrido parte superior.

El segundo ciclo se encarga de encontrar la intensidad de luz valida en la parte derecha, como se muestra en la figura 3.5.3.

```
For ejex = 233 To 195 Step -2
  For ejey = 228 To 139 Step -2
    Valor = Picture1.Point(ejex, ejey)
    Valtot = Valor
    If Valor <> -1 Then
      If Valor = 16777215 Then
       Porcen = Valtot * (Val(Combo1.Text) / 100)
       If k = 0 Then
         derecha = ejey
         Exit For
       End If
      End If
    End If
  Next ejey
```

Figura 3.5.3 Barrido parte derecha.

```
    DoEvents
  Next ejex
```

El tercer ciclo se encarga de encontrar la intensidad de luz valida en la parte inferior, como se muestra en la figura 3.5.4.

```
  For ejey = 228 To 139 Step -2
    For ejex = 233 To 195 Step -2
      Valor = Picture1.Point(ejex, ejey)
      Valtot = Valor
      If Valor <> -1 Then
        If Valor = 16777215 Then
          Porcen = Valtot * (Val(Combo1.Text) / 100)
          If k = 0 Then
            inferior = ejex
            Exit For
          End If
        End If
      End If
    Next ejex
    DoEvents
  Next ejey
```

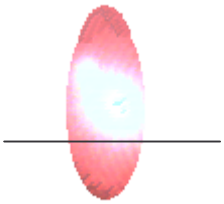


Figura 3.5.4 Barrido parte inferior.

El cuarto y último ciclo se encarga de encontrar la intensidad de luz valida en la parte izquierda, como se muestra en la figura 3.5.5.

```
  For ejex = 195 To 233 Step 2
    For ejey = 139 To 228 Step 2
      Valor = Picture1.Point(ejex, ejey)
      Valtot = Valor
      If Valor <> -1 Then
        If Valor = 16777215 Then
          Porcen = Valtot * (Val(Combo1.Text) / 100)
          If k = 0 Then
            izquierda = ejey
```

Figura 3.5.5 Barrido parte izquierda.

```
            Exit For
          End If
        End If
      End If
    Next ejey
    DoEvents
  Next ejex
```

Establecidos los 4 puntos validos se calcula el centro de la luz láser, para ello se aplican las siguientes operaciones:

ejey = Entero((inicio_derecha + inicio_izquierda) / 2)

ejex = Entero((inicio_inferior + inicio_superior) / 2)

Una vez calculado el centro como lo muestra la figura 3.5.6 se realiza un monitoreo a la intensidad de luz valida, para ello se aplica procesamiento digital de imágenes, de esta manera se puede determinar si alguien ha entrado al hogar, es decir, si la luz ha sido interrumpida la computadora va a reaccionar activando la alarma.

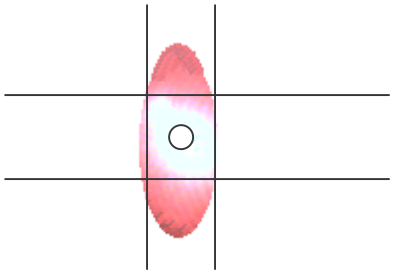


Figura 3.5.6 Centro del haz de luz.

Si la opción seleccionada en el software es la alarma local, la computadora va a reproducir mediante voz[10] un texto escrito previamente dirigido al puerto de sonido, con el propósito de que la persona desista de sus intenciones y salga de la casa una vez que se da por enterado que ha sido descubierto. Si la alarma esta en opción modo remoto, la reacción del sistema es realizar llamadas telefónicas previamente indicadas en una base de datos; y empezar a marcar, en caso de no tener éxito en la primer llamada el sistema reintentará sucesivas veces ya sea con el mismo número u otros tratando que el interesado se de por enterado.

3.6 GENERADOR DE LLAMADAS POR MODEM.

Cuando la alarma remota u otra tarea activa un procedimiento que implica utilizar la línea telefónica, el sistema se encarga de realizar una o más llamadas almacenadas[7] en su base de datos, asegurándose de que las personas involucradas han sido enteradas de los acontecimientos que han ocurrido, para lograr este fin; se utiliza un módem U.S.Robotics siguiendo una serie de pasos descritos a continuación:

a) Indicar el número de puerto a utilizar.
   Ejemplo: MSComm1.CommPort = 3
b) Establecer los parámetros de velocidad en bytes, paridad, bits de datos y bits de parada.
   Ejemplo: MSComm1.Settings = "9600,N,8,1"
c) Abrir el puerto de comunicaciones del módem.
   Ejemplo: MSComm1.PortOpen = True
d) Marcar el número utilizando para ello el comando ATDT, seguido de retorno de carro.
   Ejemplo: MSComm1.Output ="ATDT" + NUMERO + ";" + vbCr
   La instrucción vbCr equivale a Chr(13) que es: Carácter de retorno de carro
e) Si el módem devuelve “ok“ significa que la llamada esta en progreso.
f) Asegurarse que fue contestada la llamada por una persona y no por la contestadora, para ello se solicita que se presione algún digito del teléfono, si no se hace se toma como llamada no contestada.
g) Para colgar se utiliza el comando ATH, seguido de retorno de carro.
   Ejemplo: MSComm1.Output = "ATH" + vbCr
h) Finalmente se debe liberar el puerto.
   Ejemplo: MSComm1.PortOpen = False

3.7 IDENTIFICADOR DE LLAMADAS POR MODEM.

El identificador de llamadas es un servicio que ofrece la compañía telefónica y que nos permite conocer el número desde el que nos llaman antes de descolgar el teléfono. Pero no sólo eso, también trasmite otras informaciones, como son la fecha y la hora. El protocolo utilizado es el SDMF (Single Data Message Format – Formato Simple de Mensajes de Datos).

Algo importante en los módems es que no todos soportan esta función por lo que puede que una aplicación funcione en una computadora y en otra no.

Para recibir llamadas telefónicas[8] con identificación se probo en el módem U.S.Robotics siguiendo los pasos que a continuación se describen:

a) Indicar el número de puerto a utilizar.
   Ejemplo: MSComm1.CommPort = 3
b) Establecer los parámetros de velocidad en bytes, paridad, bits de datos y bits de parada.
   Ejemplo: MSComm1.Settings = "9600,N,8,1"
c) Especificar el protocolo de conexión de hardware RTS/CTS (Petición de envío/Preparado para enviar).
   Ejemplo: MSComm1.Handshaking = 2
d) Abrir el Puerto de comunicaciones.
   Ejemplo: MSComm1.PortOpen = True
e) Verificar si el módem tiene capacidad para identificar las llamadas, para lo cualse utiliza el comandoAT#CID=1.
   Ejemplo: MSComm1.Output = "AT#CID=1" + vbCr
   El comando AT#CID=1 tiene la función de checkear el módem con la finalidad de saber si se tiene la capacidad de identificar la llamada.
   Por lo que es necesario leer el manual del módem para verificar el tipo de comando correcto.
   Una vez determinado el comando el número de la persona que nos llama llega entre el primero y segundo tono de marcado.
f) Monitorear el evento MSComm1_OnComm( ), para detectar una llamada entrante.
g) Cuando llega el primer Ring se ejecuta el comando AT#CID=2.
   Ejemplo: MSComm1.Output = "AT#CID=2" + vbCr
   Entonces se recibe una cadena como la siguiente:
   La cadena se recibe en hexadecimal, el significado se describe en la tabla 3.7.1.

CADENA ENVIADA POR COMPAÑÍA TELEFÓNICA			
MESG=80160108303832343131343702 0A32343134313032303139C2			
Valor hexadecimal	Valor ASCII	Significado	
80	50	Tipo de protocolo	
16	22	Número de bytes de información sobre fecha, hora y número que se transmite.	
01	01		
08	08		
30	48	0	08/24 Día/mes
38	56	8	
32	50	2	
34	52	4	
31	49	1	11:47 Hora:Minutos
31	49	1	
34	52	4	
37	55	7	
02	02		
0A	10		
32	50	2	2414102019 Número telefónico
34	52	4	
31	49	1	
34	52	4	
31	49	1	
30	48	0	
32	50	2	
30	48	0	
31	49	1	
39	57	9	
C2	194		

Tabla 3.7.1 Parámetros del identificador de llamadas

i) Para contestar se utiliza el comando ATA.

   Ejemplo: MSComm1.Output = "ATA" + vbCr

j) Para colgar se utiliza el comando ATH, seguido de retorno de carro.

   Ejemplo: MSComm1.Output = "ATH" + vbCr

k) Finalmente se libera el puerto.

Ejemplo: MSComm1.PortOpen = False

3.8 ENVÍO DE CORREO UTILIZANDO EL PROTOCOLO SMTP.

Establecida la conexión a Internet, se envían una serie de comandos para la trasferencia de información vía e-mail.

Para establecer el enlace se utiliza el número de puerto 25 conectándose con el Host Remoto.

Ejemplo:

Socket.RemotePort = 25

Socket.RemoteHost = “smtp.prodigy.net.mx”

Una vez establecida la conexión se envía una serie de parámetros que a continuación se escriben y que son descritos en la tabla 1.2.6.1:

```
Socket.SendData = "HELO Saludos"
Socket.SendData = "MAIL FROM: <" & De & ">"
Socket.SendData = "RCPT TO: <" & Para & ">"
Socket.SendData = "DATA"
Socket.SendData = "From: " & De
Socket.SendData = "To: " & Para
Socket.SendData = "Subject: " & Asunto
Socket.SendData = "DATE: "
Socket.SendData = "MIME-Version: 1.0"
Socket.SendData = "Content-Type: multipart/mixed;"
Limite = "----=_NextPart_000_0005_01BB55BB.B5555BB5"
Socket.SendData = Chr(9) & "boundary=""" & Limite & """"
Socket.SendData = "X-Priority: 3"
Socket.SendData = "X-MSMail - Priority: Normal"
Socket.SendData = "X-MimeOLE: Producido por DiCa Ver. 1.0"
Socket.SendData = ""
Socket.SendData = "Esto es un mensaje multiparte en formato MIME"
Socket.SendData = ""
Socket.SendData = "--" & Limite
```

```
Socket.SendData = "Content-Type: text/plain;"
Socket.SendData = Chr(9) & "charset=""x-user-defined"""
Socket.SendData = "Content-Transfer-Encoding: 8bit"
Socket.SendData = ""
Socket.SendData = MensajeSaliente.Text
Socket.SendData = ""
InicioLimiteAnexo = 24
Socket.SendData = "--" & Limite
Socket.SendData = "Content-Type: application/octet-stream;"
Socket.SendData = Chr(9) & "Name=""" & FicheroAnexo & """"
Socket.SendData = "Content-Disposition: attachment;"
Socket.SendData = Chr(9) & "filename=""" & FicheroAnexo & """"
Socket.SendData = "Content-Transfer-Encoding: base64"
InicioAnexo = 30
Socket.SendData = ""
```

Si se envía un archivo anexo, éste se debe codificar en Base 64 para ello se muestra a continuación la forma de convertirlo:

```
Dim Caracter As String * 1
Dim Trio(3) As Integer
Dim Cont As Integer
Dim ContLinea As Integer
Dim Cuatro(4) As Integer
Dim Pos As Long
Dim Salir As Boolean
Dim Base64 As String
REM Se asigna el alfabeto basé 64 a la variable Base64.
Base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
REM Se abre el archivo en formato binario.
Open Direccion For Binary As #3
ContTotal = 0
REM Mientras ContTotal no llegue a al tamaño en bytes del archivo se repite el ciclo.
```

```
While ContTotal <> LOF(3)
    REM Se lee byte por byte del archivo lógico número 3.
    Caracter = Input(1, 3)
    REM Contar el número total de bytes que se leen.
    ContTotal = ContTotal + 1
    REM Contar el número de bytes leídos dónde el máximo es 3 y reinicia la cuenta.
    Cont = Cont + 1
    Trio(Cont) = Asc(Caracter)
    If Cont = 3 Then
      Cuatro(1) = Int(Trio(1) / 4)
      Cuatro(2) = (Trio(1) - Int(Trio(1) / 4) * 4) * 16 + Int(Trio(2) / 16)
      Cuatro(3) = (Trio(2) - (Int(Trio(2) / 16) * 16)) * 4 + Int(Trio(3) / 64)
      Cuatro(4) = Trio(3) - Int(Trio(3) / 64) * 64
      Cont = 0
      ContLinea = ContLinea + 4
      cadena = cadena & Mid(Base64, Cuatro(1) + 1, 1) & Mid(Base64, Cuatro(2) + 1, 1) &
      Mid(Base64, Cuatro(3) + 1, 1) & Mid(Base64, Cuatro(4) + 1, 1)
      If ContLinea = 76 Then
        Salir = True
        t = Len(cadena)
      End If
    End If
  Wend
  Close #3
  REM En caso de que la última lectura sea de 1 ó 2 bytes se aplica la siguiente operación
  If ContTotal=LOF(3) then
    Case 1
          Cuatro(1) = Int(Trio(1) / 4)
          Cuatro(2) = (Trio(1) - Int(Trio(1) / 4) * 4) * 16 + Int(Trio(2) / 16)
          cadena = cadena & Mid(Base64, Cuatro(1) + 1, 1) & Mid(Base64, Cuatro(2) + 1,
          & "=="
```

```
    Case 2
        Cuatro(1) = Int(Trio(1) / 4)
        Cuatro(2) = (Trio(1) - Int(Trio(1) / 4) * 4) * 16 + Int(Trio(2) / 16)
        Cuatro(3) = (Trio(2) - (Int(Trio(2) / 16) * 16)) * 4 + Int(Trio(3) / 64)
        cadena = cadena & Mid(Base64, Cuatro(1) + 1, 1) & Mid(Base64, Cuatro(2) + 1,
         & Mid(Base64, Cuatro(3) + 1, 1) & "="
    End Select
End if
```

3.9 LECTURA DE CORREO UTILIZANDO EL PROTOCOLO POP3.

Para leer el correo electrónico que es asignado por el proveedor de Internet al realizar el contrato por el servicio.

Para establecer el enlace se utiliza el número de puerto 110 conectándose con el Host Remoto.

Ejemplo:

```
Socket.RemotePort = 110 'Puerto donde regularmente esta el servicio de POP
Socket.RemoteHost = "pop.prodigy.net.mx"  'Servidor POP
```

Una vez establecida la conexión se envía una serie de parámetros que a continuación se escriben:

```
Socket.LocalPort = 0 'Si se asigna cero el control Winsock automáticamente da un número de
                     'puerto
Socket.Connect
'Mandar el nombre de usuario
  wnsWinsock.SendData "USER " & Usuario & vbCrLf
'Mandar el password
Socket.SendData "PASS " & Contr & vbCrLf
'Una ves que el Usuario y Contraseña son confirmados, se ve cuantos mensajes hay
Socket.SendData "LIST" & vbCrLf
```

BIBLIOGRAFÍA.

[1] CEBALLOS SIERRA FCO. JAVIER. Visual C++ 6 Programación Avanzada en Win32. ED. Alfaomega, RA-MA. 1ra. ed. México 1999. pp. 460-500.
[2] Acceder al puerto paralelo en Visual Basic. [Consulta 11 febrero 2003]. Formato HTML. http://www.idg.es/pcworld/ShowSol.asp?ID=458
[3] CFrameGrabber - Wrapper for AVICap Window. [Consulta 13 febrero 2003]. Formato SHTML.
http://www.codeguru.com/multimedia/AviCapWrp.shtml
[4] AviCap32.dll. [Consulta 13 febrero 2003]. Formato HTML.

http://www.experts-exchange.com/Programming/Programming_Languages/Visual_Basic/Q_20830183.html
[5] CEBALLOS SIERRA FCO. JAVIER. Curso de programación de Visual Basic 6. ED. Alfaomega, RA-MA. 1ra ed. México 2000. Pág. 277.
[6] NATHAN GUREWICH /ORL GUREWICH. Aprendiendo Visual Basic 5 en 21 días. ED. Prentice-Hall. 1ra ed. México 1998. pág. 280.
[7] CARBALLAR JOSÉ A. Comunicaciones del PC. ED. Alfaomega, RA-MA. 2da. Ed. México 1997, pp. 388-410.
[8] Caller ID. [Consulta 15 febrero 2003]. Formato HTM.
http://www.esi2.us.es/lacomunateleco/web/cacharreo/callerID_c.htm
[9] http://www.tek-tips.com/gviewthread.cfm/lev2/4/lev3/32/pid/710/qid/607524
[10] Texto-Voz. [Consulta 16 febrero 2003]. Formato HTM.
http://www.funcaragol.org/html/fitecsvb.htm

CAPÍTULO IV

PRUEBAS Y RESULTADOS

4.1 INTRODUCCIÓN.

Las pruebas realizadas al sistema en general y los resultados obtenidos se presentan para conocer el entorno en que se desarrolla el proyecto. Las ventajas de este sistema son positivas y facilita las comunicaciones ya que hoy en día la información es numerosa y llega muchas veces de forma imprevisible. Eso quiere decir que el dato que llega es procesado inmediatamente siempre con un control multitarea.

Siendo necesario una gran cantidad de pruebas con la finalidad de mejorar los tiempos de procesamiento para obtener resultados lo más rápido posible y sin interferir en otras tareas que en ese mismo momento se estén procesando.

Las pruebas del sistema tanto de hardware como de software se dividieron en tres fases:

a) Establecimiento. Durante la cual, por un lado, se establece la conexión física de la computadora, ya sea con la cámara web, DTMF o línea telefónica, y mediante software se hace el intercambio de información.
b) Transmisión de información. En la que interfaces y computadora intercambian datos a través del enlace establecido. Durante esta fase, se lleva a cabo la comprobación de la información transmitida y/o recibida para que en el caso de que exista un error ya sea por parte del periférico o del programa poder detectarlo y corregirlo.
c) Terminación. En la que se da por terminada la comunicación.

En los siguientes puntos se muestran los resultados más importantes, obtenidos a través de diversas pruebas.

4.2 HARDWARE.

4.2.1 ALARMA LÁSER.

Al realizar pruebas para comprobar la distancia alcanzada con el rayo láser utilizando la configuración mostrada en la figura 2.4.1, el perímetro a proteger se veía altamente restringido ya que sólo llegó a cubrir una distancia de 70 metros, esto debido a la poca potencia que mostró la energía extraída del puerto paralelo (5 V. de corriente directa). Para aumentar el alcance se modifico la conexión agregando una entrada de corriente con una pila de 9V. y un relevador de 6V.

Una vez realizada la conexión mostrada en la figura 2.4.2, se logro aumentar considerablemente la potencia del láser, incrementando 5.7 veces el alcance. Cubriendo un perímetro máximo de 400 metros, creando así un modelo que mejora considerablemente la protección de una mayor área.

Mejorado el alcance de protección, se activó la alarma y se inicio una constante supervisión de la luz láser para detectar su interrupción, el tiempo de supervisión se realizó cada milisegundo ratificando así una sensibilidad alta.

Las siguientes fotografías muestran la forma en que se implementa la protección en una casa mediante rayo láser.

Figura 4.2.1.1 Sistema de seguridad por reflexión.

4.2.2 TIMBRE POR PUERTO PARALELO LPT2.

El experimento para detectar cuando una persona oprime el botón del timbre consistió en conectar el pin 25 y el pin 8 al botón pulsador. Previamente se debe energizar el puerto paralelo; sin embargo esto se logra enviando a la dirección 278H el valor decimal de 255, una vez energizados los 8 pines al oprimir el timbre es enviada tierra al pin 8 lo que provoca que se apage el pin y sea detectado el toque de tiembre, activando de esta forma la cámara web y fotografiando a la persona que se encuentre oprimiendo el botón del timbre tal y como lo muestra la figura 4.2.2.1.

4.2.2.1 Fotografía almacenada al tocar el timbre.

4.2.3 DECODIFICADOR DTMF.

Durante los experimentos realizados con el circuito M-8870-01 mostrado en la figura 2.5.1, se percibió la presencia de ruido al momento de contestar el teléfono con la computadora o desde cualquier otro teléfono conectado a la línea, provocando en muchos casos que no se entendiera la conversación. La alternativa para eliminar el ruido y dar solución al problema presentado se logro implementando al circuito original un transformador de acoplamiento telefónico de 600 Ω, quedando la configuración final como se muestra la figura 2.5.2.

Una vez que el circuito mostró estabilidad se conecto al puerto LPT1 con dirección lógica 379H, y mediante la función de *Lectura* de la librería puerto.dll se verifico el valor obtenido del puerto paralelo cuyo valor fue de 0 a 15.

Una de las primeras pruebas realizadas para verificar el buen funcionamiento tanto del software como del DTMF se muestra en la figura 4.2.2.1, este antecedente se hizo con la finalidad de verificar que el puerto LPT1 recibía información.

En los siguientes incisos se describen las pruebas realizadas:

a) Se conectó el pin 25 al pin 10 del puerto paralelo y realizó una lectura del puerto 379H. Obteniendo como respuesta un valor de 255.
b) Posteriormente el pin 25 al pin 11 y se lee el puerto 379H. Y así con los demás pines. Obteniendo como respuesta un valor de 64.
c) Pin 25 con pin 12 y se leyó el puerto 379H. Obteniendo como respuesta un valor de 32.
d) Pin 25 con pin 13 y se leyó el puerto 379H. Obteniendo como respuesta un valor de 16.
e) Pin 25 con pin 15 y se leyó el puerto 379H. Obteniendo como respuesta un valor de 8.

Una vez que se verificó mediante un procedimiento de lectura al puerto el cambio de valor de cada pin se probo el circuito de la figura 2.5.3, pulsando dígitos del teclado telefónico y comprobando el valor obtenido en el rango de 0 a 15.

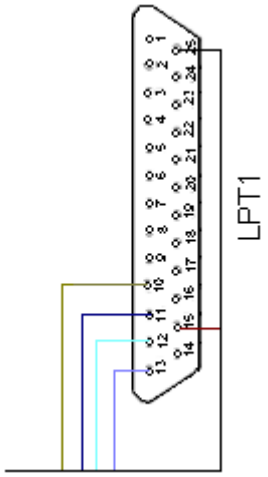


Figura 4.2.3.1 Verificación pines puerto paralelo.

4.3 SOFTWARE.

4.3.1 PUERTO PARALELO.

La librería creada en Visual C++ 6.0 funcionó correctamente al interactuar con el puerto paralelo durante las pruebas realizadas en Visual Basic, no presentando ningún problema en las funciones *Escritura* y *Lectura*, ejecutándose de forma local y global, obteniéndose los resultados esperados. La operación de *Lectura* al puerto con dirección 379H se aplico cada 50 milisegundos, demostrando este lapso de tiempo como el optimo para la detección en la pulsación de un digito del teclado telefónico.

4.3.2 CÁMARA WEB.

La obtención de video por medio de una cámara web resultó satisfactorio utilizando la librería avicap32.dll, las pruebas realizadas se produjeron con una cámara web Alaska y posteriormente con una Sceptre obteniendo la transmisión de video en el objeto Form1 con ambas cámaras.

La lectura de frames proveniente de las cámaras se realizó cada 66 milisegundos, considerando esa una velocidad aceptable para el procesamiento de la imagen.

4.3.3 ALARMA LÁSER.

Obtenida la imagen del láser como se muestra en la figura 3.5.1 mediante 6 espejos y cubriendo un área de 32 metros se reflejo el rayo cubriendo todos aquellos puntos considerados como sensibles. Finalmente la reflexión termina en un lugar visible por la cámara web, los resultados fueron que al rededor de la luz se observa un color rojizo y el centro es percibido como un blanco intenso siendo estos elementos los utilizados para determinar si el láser ha sido enfocado por la cámara web y validado por el software durante el procesamiento de la imagen.

Durante las pruebas realizadas se determinó que el lugar donde es captado el láser por la cámara web no debe ser junto a una ventana, debido a que el viento puede penetrar por rendijas y obligar a la cortina o persiana a estar en continuo movimiento; llegando a provocar una falsa alarma por la variación en la intensidad de luz proveniente del exterior. Además que existe la posibilidad que el sol entre directamente a la cámara web, lo que provocaría la inhabilitación de la seguridad ya que la intensidad de la luz del sol sería equivalente a la luz proporcionada por el rayo láser, pudiendo interrumpirse el rayo sin ser percibido por la cámara web.

4.3.4 GENERADOR DE LLAMADAS POR MODEM.

Al activarse la alarma externa, el sistema se conecta a la línea telefónica mediante módem marcando al primer número telefónico indicado en la base de datos. El tiempo que tarda conectarse a la línea telefónica y marcar al número telefónico es de 11 segundos. Si el número al que marca se encuentra ocupado lo intenta por 10 segundos más, si durante esos intentos no se obtiene la llamada, el sistema marca al siguiente número registrado en la base de datos. En caso de que la llamada sea contestada se proporcionan 10 segundos para que el sistema detecte que el teléfono remoto ha sido descolgado por una persona y no por una contestadora, esto se detecta al solicitarle al usuario que presione un digito.

4.3.5 RECEPTOR DE LLAMADAS TELEFÓNICAS CON IDENTIFICADOR DE LLAMADAS.

Mediante el comando ATA descrito en la tabla 1.2.4.1 es posible que la computadora se comporte como una contestadora proporcionando los mismos resultados que un teléfono digital ordinario, para saber el número telefónico del que proviene la llamada; al segundo ring se manda el comando AT#CID=2. Esto provoca que el módem reciba información referente a la llamada entrante. Las pruebas se realizaron con dos módem U.S.Robotics con variación en el modelo y tipo de conexión ya que uno se conectaba al puerto serie y otro al puerto USB.

Durante el experimento realizado se detecto que no todos los módems utilizan el mismo comando, ya que cada modelo del mismo fabricante o de otro fabricante emplea distintos comandos, por lo que se sugiere consultar el manual del módem. Sin embargo los siguientes comandos indican que se requiere información adicional de la llamada entrante:

AT%CCID=1

AT+VCID=1

AT#CC1

AT*ID1

AT#CID=1

Al recibir el segundo RING se envía nuevamente alguno de los siguientes comandos dependiendo el fabricante y modelo pero ahora asignando un valor de dos:

AT%CCID=2

AT+VCID=2

AT#CC2

AT*ID2

AT#CID=2

El módem que se utilizó fue el de conexión USB y recibió la cadena enviada por la compañía telefónica en un formato más legible para el usuario final, depositando la información en el puerto de la siguiente manera:

La palabra reservada DATE seguida de un símbolo igual, y presentando primero el año seguido del mes.

DATE=0311

La siguiente palabra reservada enviada al módem es TIME seguido del símbolo igual y siendo el primer parámetro la hora seguido de los minutos.

TIME=1220

La última palabra reservada enviada por el módem es NMBR seguido del símbolo igual y a continuación el número telefónico.

NMBR=2414102019.

Siendo éste ultimo un formato más legible para el usuario y evitando una serie de conversiones como se mostró en la tabla 3.7.1. y que generó el módem U.S.Robotics con conexión al puerto serie.

4.3.6 ENVÍO DE CORREO UTILIZANDO EL PROTOCOLO SMTP.

Al enviar información vía e-mail el sistema tarda en conectarse al servicio de Internet 55 segundos, las direcciones a las que se enviará texto/archivos previamente se indican en la base de datos del propio sistema. Durante el envió de información se incluyeron archivos de diferente formato tales como txt, doc, bmp, jpg, xls sin ningún problema. Cabe mencionar que los archivos o mensajes sólo pueden a enviarse a la cuenta proporcionada por la compañía que provee el servicio de Internet.

CAPÍTULO V

CONCLUSIONES

5.1 CONCLUSIONES.

Con la propuesta se obtuvo un administrador de información factible y un medio de respuesta inmediata sobre sucesos acontecidos principalmente durante la ausencia de los integrantes de la casa, recopilando grandes cantidades de información para su posterior consulta.

La obtención de la información guardada se realizó mediante una llamada telefónica a la vivienda, indicándole al sistema mediante la pulsación de tonos que proporcionara datos de las personas que se han comunicado al hogar, y respondiendo audiblemente el con el número telefónico o nombre de la persona, la hora y la fecha de la llamada.

Asimismo se ordenaron remotamente una serie de operaciones con un mínimo de tiempo; tal fue el caso de indicarle al sistema que en determinada hora enviara automáticamente la información almacenada hasta ese momento a una o más direcciones de e-mail, logrando de esta manera consultar las fotografías, la hora y la fecha de las personas que habían tocado el timbre.

El sistema también incluyó acciones de respuesta inmediata, realizándose cuando la seguridad implementada mediante rayo láser se veía interrumpida; y respondiendo a tal evento mediante una o más llamadas telefónicas especificadas previamente; dando aviso del problema acontecido.

Cuando los integrantes de la casa se encentraban en ella el sistema funciono alertando al instante audible y visualmente sobre la entrada de una llamada telefónica, presentando información de la persona que estaba intentando comunicarse con alguno de los integrantes, y dejando que el usuario tomara la decisión de contestar, o ignorando la alerta para que la computadora se encargara de la acción correspondiente. Si la acción acontecida es el toque del timbre de la puerta principal, el sistema automáticamente presenta video, para que los integrantes identifiquen a la persona.

Finalmente con el desarrollo del proyecto se logró presentar la unión de diversas aplicaciones de la informática y la electrónica para conformar un innovador medio digital de comunicación.