UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE SEGURIDAD CONTROLADO VÍA UN TELÉFONO CELULAR.

TESIS PROFESIONAL
para obtener el título de
INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTA:
ALEJANDRO JESÚS ADAME GUTIÉRREZ

DIRECTOR DE TÉSIS:
M. en Ing. Larry H. Escobar Salguero

Ciudad Universitaria, México. Marzo 2013.

AGRADECIMIENTOS

A mis padres.

Por todo su apoyo, sus consejos, sus valores, la motivación que me han brindado siempre y su ejemplo de perseverancia.

Al M. en Ing. Larry H. Escobar Salguero.

Por su infinita paciencia durante la elaboración de este trabajo de tesis, por su trato siempre cordial a pesar de todas las adversidades y por brindarme siempre su apoyo y amistad.

Al M. en C. Ranulfo Rodríguez Sobreyra.

Por su valiosa contribución y apoyo al final de este largo proceso de titulación.

A mi amiga Nelly Adriana Amador García.

Por todo su apoyo a lo largo de este proceso de titulación, por compartir tanto tiempo conmigo y por brindarme su invaluable amistad siempre.

PREFACIO.

El presente trabajo describe el diseño e implementación de un sistema de seguridad para viviendas o inmuebles que es controlado a través de una llamada telefónica utilizando un teléfono celular. Para este proyecto se investigó acerca de algunos puertos e interfaces de comunicación de una PC con dispositivos externos. Se brindan algunos detalles sobre lenguajes de programación del hardware utilizado y se extienden a algunos ejemplos sencillos relacionados con la implementación llevada a cabo en este trabajo.

Se parte de un estudio de las estructuras básicas de estos sistemas hasta llegar a plasmar la composición final del sistema implementado. Se describe el funcionamiento de cada uno de los módulos que conforman el sistema y la manera en que se conectan y comunican entre sí dichos dispositivos. Se analizan y detallan todos los conceptos teóricos del procesamiento digital de señales que se aplican dentro del sistema de seguridad. Se muestran no sólo los diagramas de flujo de los algoritmos utilizados, sino también los códigos que representan dichas implementaciones en lenguaje C o Visual Basic según correspondiera el algoritmo.

La mayor parte de los bloques del sistema, más específicamente los sensores y periféricos de entrada y salida se simplificaron todo lo posible en su diseño para facilitar su posterior implementación. Se detallaron las cuestiones técnicas y los resultados del comportamiento que tuvieron estos elementos, sin embargo no se profundizaron en dichas cuestiones porque el objetivo de este trabajo fue más enfocado en la cuestión del procesamiento de las señales y los algoritmos de control del sistema. Es importante mencionar que la complejidad de un sensor puede resultar determinante o fundamental en el comportamiento de un sistema de cualquier tipo, y por ende es algo que no debe menospreciarse, para ahondar en el tema habría la forzosa necesidad de entrar en el estudio de la medición e instrumentación, lo cual no es el objetivo particular de este proyecto de tesis.

Se hace un énfasis especial en detallar el módulo de procesamiento y control así como los algoritmos de toma de decisiones del sistema, esto debido a que forman la parte medular del sistema. Continuando con el mismo orden se plantearon las formas más simples de implementar los diversos dispositivos y periféricos de salida.

Para concluir, se muestran los diagramas esquemáticos de la circuitería implementada y todas las conexiones entre dispositivos se describen tal como fueron hechas para el prototipo diseñado en este proyecto.

ÍNDICE.

CAPÍTULO 1.
INTRODUCCIÓN.

En muchas ocasiones es necesario vigilar el funcionamiento e interactuar con sistemas de diversos tipos en tiempo real y de forma remota, por lo tanto tenemos que recurrir a otros servicios y sistemas que nos posibiliten dicha comunicación de la manera más rápida, eficiente y segura. Por otra parte uno de los problemas que más aquejan al país es el alto nivel de inseguridad, debido a esto cada vez es más necesario contar con elementos que nos ayuden a proteger nuestra integridad personal y familiar, así como los bienes que poseemos. Existen compañías que brindan servicios para ayudar a atacar dicha problemática, sin embargo éstos muchas veces sobrepasan las posibilidades económicas de la población en general, haciéndolos prácticamente inalcanzables para las personas de clase media y baja.

Analizando los problemas antes mencionados se hace evidente la necesidad de contar con sistemas de seguridad económicos, y que posean un medio de comunicación con el usuario que permita el control e interacción con el mismo para tratar de garantizar el bienestar de las personas y sus bienes.

Cualquier sistema de seguridad electrónico se compone de tres partes fundamentales:

- Detección. Ésta se realiza por medio de sensores de diversos tipos, los cuales puedan captar ciertas amenazas o eventos de interés para el usuario.
- Central de procesamiento y control. En donde se reciben y procesan las señales de los sensores y con base en ello se toman decisiones que pueden involucrar al resto de los periféricos del sistema.
- Periféricos de salida. Tales como luces, sirenas, cámaras, teléfonos, etc.

Como se ha mencionado, uno de los periféricos del sistema puede ser un teléfono fijo o móvil, en la mayoría de los casos, éste se utiliza para informar a la central de monitoreo o al propietario del sistema que algún evento de interés se ha llevado a cabo. Sin embargo, la llamada telefónica puede ser aprovechada también para enviar diversas opciones del sistema, de modo que sea posible interactuar con el mismo de forma remota. Esto se puede lograr por medio del procesamiento del audio de la llamada, ya sea por medio de la voz, o bien, una opción más sencilla es por medio de *tonos duales de multi-frecuencia* (DTMF) del teclado telefónico.

Objetivo: Diseñar e implementar una interfaz que permita interactuar con un sistema de seguridad para casa habitación por medio de una llamada telefónica utilizando DTMF. El sistema deberá ser capaz de detectar cualquier intromisión o amenaza en la propiedad, por medio de diversos sensores colocados en las puertas y ventanas, de igual forma verificará si se ha accionado el timbre de la propiedad.

Las señales enviadas por los sensores y el timbre serán recopiladas por un microcontrolador, el cual transmitirá dicha información en un paquete de datos a una PC a través de un puerto USB. Con base en el paquete de datos recibido, la PC enviará diferentes comandos al microcontrolador por USB y a un teléfono móvil vía Bluetooth.

El microcontrolador también deberá ser capaz de detectar la presencia de tonos DTMF en el audio del teléfono móvil, una vez muestreado el mismo, enviará los datos por el puerto USB para que la PC los procese, ésta realizará una transformada discreta de Fourier (DFT) con los datos adquiridos y obtendrá como resultado el espectro del audio muestreado, de esta forma conocerá o identificará las componentes en frecuencia y se podrá saber si se ha presionado alguna tecla telefónica y de ser así cuál de ellas se presionó. Una vez establecida la conexión con el sistema por medio de una llamada telefónica, se podrán mandar comandos, introducir contraseñas, etcétera, por medio de códigos formados por las teclas telefónicas, ayudados además con menús auditivos proporcionados por el mismo sistema a la hora de realizar la conexión.

El método antes mencionado se implementará utilizando una PC en la cual se ha instalado una aplicación en Visual Basic que administra el estado y el tránsito de las señales de los sensores y el resto de los periféricos de los que dispone el sistema de seguridad. Dichas señales serán recibidas por un microcontrolador PIC18F4550, el cual presenta facilidades para establecer comunicación USB con otros dispositivos, entonces el microcontrolador adecuará, recopilará y multiplexará las señales para que sean transmitidas por el puerto USB hacia la PC tal como se muestra en la figura 1.

Como ya se ha mencionado, una de las principales funciones del sistema es poder realizar comunicación con el usuario por medio de una llamada telefónica, esto además con el objetivo de comunicar cierto estado del sistema, si se ha realizado algún evento de determinado interés para el usuario, o bien, mandar diversos comandos al sistema, de forma que el usuario pueda tener acceso a ciertos procesos de control de forma remota. Un ejemplo de esto último puede ser inclusive la propia activación y desactivación del sistema entero, o el control de accesos a la propiedad.

La comunicación telefónica se establece y controla por la PC mediante comandos AT, éstos son comandos que permiten la interacción con un modem telefónico y son denominados así por abreviatura de la palabra inglesa "*attention*". El ambiente Visual Basic nos provee las herramientas necesarias para establecer dicha conexión, ésta se establece por medio del puerto Bluetooth empleándose como un puerto serie virtual. En la figura 1 se muestra un diagrama estructural del sistema en general.

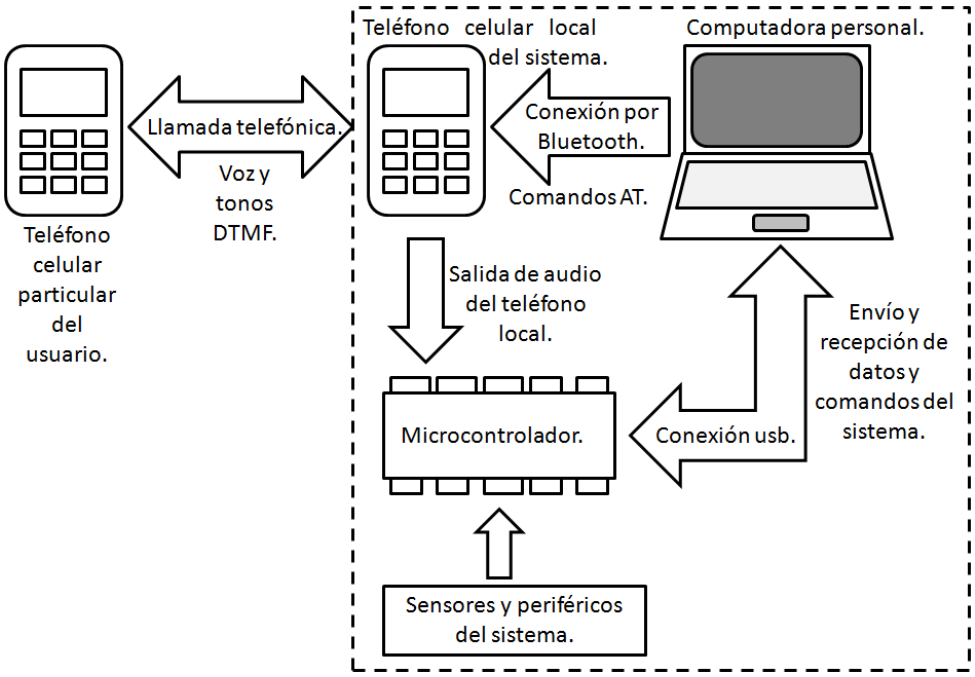


Figura 1. Diagrama general del sistema.

A futuro, un método alternativo de control para dicho sistema es el procesamiento de voz, estableciendo de igual forma una llamada telefónica, pero en este caso realizando comandos de voz en lugar de códigos con base en tonos. De esta manera el control del sistema podría ser aún más personalizado y por ende más sofisticado. Sin embargo, para aumentar la sofisticación del sistema es necesario emplear dispositivos más eficientes, con costos más elevados que hacen al sistema menos accesible.

Otro método empleado comúnmente en diversos sistemas no solamente de seguridad es establecer una comunicación por medio de la red de internet. Hoy en día existen dispositivos móviles que poseen conexión a internet en casi cualquier lugar en donde se encuentre el usuario, sin embargo, estos dispositivos no tienen un costo accesible para toda la población y además también presentan un mayor riesgo a infiltraciones de usuarios no autorizados debido a que obviamente la red de internet es mucho más abierta que la red interna establecida por una compañía telefónica privada. De esta manera el sistema de seguridad para una casa habitación o negocio, tendría que ser complementada con un excelente sistema de seguridad informática de forma que evite cualquier intento de intromisión.

El proceso de desarrollo del proyecto que se describe en esta tesis se divide en varios bloques requeridos para el completo entendimiento del sistema. En el *capítulo 2* se tratarán las los fundamentos básicos de los puertos de comunicación que se pueden presentar en un equipo de cómputo personal. Se explica brevemente el funcionamiento de los puertos Bluetooth, USB y RS-232, así como algunos protocolos de comunicación de los mismos, además se explicará a grandes rasgos el funcionamiento del modem telefónico y el de algunos comandos AT que sirven para la interacción con dicho dispositivo.

En el *capítulo 3* se hará una breve descripción de los lenguajes de programación que emplearemos en el proyecto, tanto para la PC como para el microcontrolador, poniendo particular énfasis en los elementos y herramientas que utilizaremos en cada uno de ellos. Se revisarán primordialmente las sintaxis y propiedades de los lenguajes “Basic” y “C”, para proceder a complementarlos en ambientes más específicos como son Microsoft Visual Basic 6.0 y CCS Compiler.

En el *capítulo 4* se comenzará a estructurar el sistema de seguridad con fundamento en todos los conceptos teóricos que se necesiten para el diseño, especialmente en el análisis de circuitos tanto analógicos como digitales y de igual forma en el procesamiento de señales. Además se revisarán las características que presenta el microcontrolador PIC18F4550 y las funciones de comunicación que éste posee. Se mencionarán además los detalles generales de los periféricos que se pretenden emplear en el sistema.

En el *capítulo 5* se iniciará con el diseño del sistema y la implementación del mismo, para esto se realizará una descripción de los procesos de ensamblado tanto de hardware como de software para cada bloque que conforma el sistema de seguridad. Se analizarán los algoritmos de detección de sensores y tonos DTMF y se detallará la forma de configurar los elementos necesarios para la comunicación entre la PC, el microcontrolador y el teléfono celular. También se detallarán las conexiones físicas de los sensores empleados y los periféricos de salida.

Finalmente en el *capítulo 6* se plasmarán, a detalle, los resultados de las pruebas pertinentes, para posteriormente poder evaluar la efectividad del sistema y poder confirmar si se cumple fielmente con el objetivo que se propuso para el mismo.

CAPÍTULO 2.
PUERTOS DE COMUNICACIÓN EN UNA PC.

Conforme avanza el tiempo hemos podido observar las grandes mejoras en eficiencia y versatilidad de los equipos de cómputo. Tan sólo los periféricos de entrada y salida han evolucionado de cierta simplicidad a dispositivos mucho más complejos que nos facilitan la interacción con la computadora y nos brindan además servicios que inicialmente estos equipos no podían dar. La tendencia tecnológica actual nos muestra entonces que en un futuro muy cercano, además de lo que estamos acostumbrados a realizar en una PC, una gran cantidad de los servicios, aplicaciones y pasatiempos requeridos por las personas serán administrados por nuestros sistemas computacionales, y dentro de éstos, el sistema de seguridad de nuestros hogares y negocios.

Algunos de los elementos de los equipos de cómputo que más han evolucionado a lo largo del tiempo son los periféricos de entrada y salida. En un principio por ejemplo, para poder introducir información o programas a una computadora se debía hacer uso de tarjetas perforadas codificadas de tal forma que la computadora recibiera dicha información, hoy día estamos acostumbrados a utilizar el teclado para realizar esta función e inclusive en algunos sistemas ya se utilizan pantallas táctiles, o también se emplean sistemas de procesamiento de voz y sonidos para tal efecto, de forma que el teclado común empieza a ser obsoleto.

En este capítulo se detallarán las características básicas de los puertos RS-232, USB y Bluetooth que permiten crear interfaces de comunicación con un equipo de cómputo. También se describirá el funcionamiento del modem telefónico y los comandos que permiten la interacción con éste.

2.1. SISTEMAS DE COMUNICACIONES EN UNA PC.

Recordando la estructura de un sistema de cómputo, para que la computadora pueda interactuar con los periféricos, se necesita una vía de comunicación, por donde pueda realizarse la transferencia de información entre el CPU y los periféricos. Para esto, el sistema de cómputo cuenta con lo que llamamos "puertos". Los puertos son de entrada, de salida, o pueden tener ambas funciones a la vez, lo que dependerá del tipo de dispositivo que se conecte a dicho puerto. Los puertos de las computadoras también han evolucionado a lo largo del tiempo, mejorándose continuamente en busca de una mayor rapidez, eficiencia y comodidad.

Para el objetivo de esta tesis es necesario estudiar los puertos que utilizaremos en la comunicación de los periféricos e interfaces de nuestro sistema, así como los protocolos de los que disponen para poder realizar dicha comunicación. Los puertos utilizados en el proyecto son el USB y el Bluetooth, sin embargo es necesario detallar también el funcionamiento del puerto RS-232 debido a que utilizaremos protocolos y archivos de biblioteca que permiten modelar la comunicación de esos puertos como si fungieran como puertos seriales del tipo RS-232.

2.2. EL PUERTO SERIE RS-232.

Como ya se ha mencionado, a lo largo de los años se han desarrollado diversas interfaces estándar para comunicar un equipo de cómputo con diversos periféricos utilizando puertos de comunicación serie, entre éstos estándares se encuentran el *RS-232*, el *RS-422* y el *RS-485*. De ellos el más utilizado es sin duda el RS-232 cuyas siglas provienen de: *Recommended Standard number 232*. La actual versión de este estándar se encuentra designada como *TIA/EIA-232C* ya que se encuentra registrada por la *Telecommunications Industry Association* (TIA) [7].

Antes de la llegada del USB, este puerto se utilizaba fundamentalmente como conexión del mouse de una computadora. Sin embargo su protocolo de comunicación también es vital para establecer la conexión entre un equipo de cómputo y un modem telefónico, y es bastante útil y cómodo para monitorear y comunicar datos de medición e instrumentación con diversos dispositivos, como podría ser por ejemplo un microcontrolador. Algunas características importantes de la comunicación serie con un puerto RS-232 son [1]:

- Soporta protocolos de comunicación serie tanto síncrono como asíncrono.
- Soporta un número máximo de dos dispositivos conectados entre sí.
- La velocidad de comunicación común es de aproximadamente 20 Kbits/seg, sin embargo con ciertos dispositivos, como un modem por ejemplo, puede llegar a ser de hasta 115 Kbits/seg.
- Los datos pueden ser manejados en código binario o como texto en código ASCII.

La figura 2.1 muestra el conector más utilizado para la conexión RS-232, el que está conformado por nueve pines (DB-9), aunque también existe el conector de veinticinco pines (DB-25). El protocolo primario de comunicación es llevado a cabo únicamente mediante el uso de tres pines del conector, el pin de transmisión de datos (TD), el pin de recepción de datos (RD) y el pin de referencia a tierra (GND). Los demás pines están disponibles para los protocolos de control de flujo de datos, pero pueden no utilizarse según la aplicación y el dispositivo [7].

Como se verá en los capítulos 4 y 5, toda la comunicación entre los elementos del proyecto presentado por ésta tesis se fundamenta básicamente en modelos que simulan la comunicación serie RS-232, de ahí la importancia de mencionar las características de este puerto.

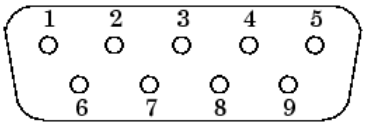


Figura 2.1. Conector RS-232 (DE-9).

El protocolo de comunicación RS-232 define a los dos dispositivos conectados mediante un puerto serie como el *Data Terminal Equipment* (DTE) y el *Data Circuit-Terminating Equipment* (DCE). Esta terminología en la comunicación RS-232 se basa en los orígenes del mismo, en el cual el estándar era implementado fundamentalmente para la comunicación entre un equipo de cómputo, que en este caso funge como DTE, y un modem telefónico, que funge como DCE. La figura 2.2 muestra un diagrama básico de la comunicación serie entre un DTE y un DCE por medio de los pines TD y RD del puerto serie [7].

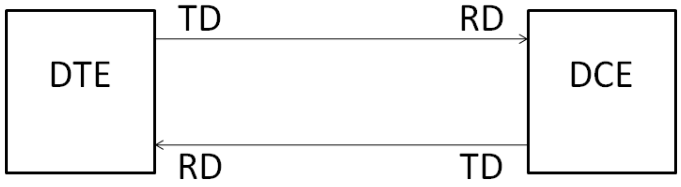


Figura 2.2. Diagrama básico de la comunicación estándar RS-232.

El formato en el que los datos son transmitidos es con paquetes de doce bits llamados "*palabras*", sin embargo, no todos los bits de una *palabra* pueden tener relevancia, esto dependerá si la comunicación es síncrona o asíncrona, y si se está empleando algún tipo de control de flujo de datos. La figura 2.3 muestra un diagrama de dicho formato:

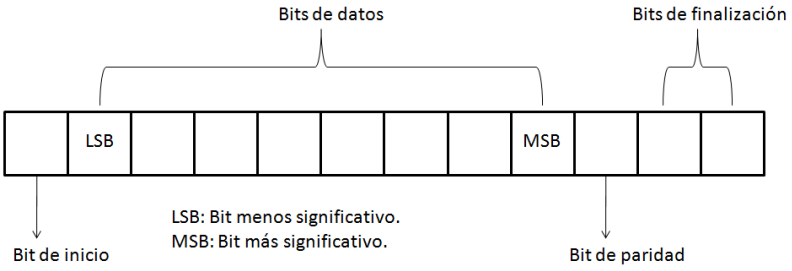


Figura 2.3. Formato de transmisión de datos del puerto RS-232.

Cuando la comunicación es *síncrona*, los bits de inicio y finalización pueden no tomarse en cuenta debido a que la transmisión de datos está controlada por una señal de reloj sincronizada tanto en el DTE como en el DCE, como consecuencia de esto, la velocidad en la comunicación es mayor que la presente en la comunicación asíncrona. La desventaja que posee es que no siempre es posible o sencillo sincronizar la señal de reloj en los dispositivos que se desea comunicar.

Cuando la comunicación es *asíncrona*, los bits de inicio y finalización son vitales para la transmisión de datos, ya que delimitan el tamaño de los mismos y controlan el propio proceso de comunicación entre los dos dispositivos. El bit de paridad sirve para algoritmos de detección y corrección de errores durante el proceso de transmisión de datos, por lo que no siempre es tomado en cuenta.

En muchos dispositivos la comunicación puede configurarse para que los bits de datos sean referidos a un carácter, debido a que estos bits pueden representar un símbolo alfanumérico en código ASCII. El orden en que los bits son transmitidos sigue los siguientes pasos:

1. Un bit de inicio es transmitido con un valor de “0”.
2. Ocho bits de datos son transmitidos. El primero representa al bit menos significativo (LSB) y el último al más significativo (MSB).
3. Un bit de paridad (si está definido) se transmite.
4. Uno o dos bits de finalización son transmitidos, cada uno con valor de “1”.

Puede notarse, que cada paquete de bits de datos puede contener hasta 256 valores diferentes, de 0 a 255 en formato binario, y para poder representar valores mayores, por ejemplo en formato *int32*, se necesitarán transmitir de forma continua (es decir, sin finalización o interrupción) cuatro *palabras* a la vez [7].

2.3. EL PUERTO SERIE USB.

En los años 90 un grupo de siete empresas, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC y Philips, diseñaron e introdujeron en el mercado al puerto de comunicación *Universal Serial Bus* (USB), esto revolucionó la comunicación y el desarrollo de hardware establecido para una computadora, ya que simplificó de manera sustancial la instalación y configuración de diversos periféricos para un sistema computacional. Antes toda PC poseía puertos de comunicación de diversa índole para poder conectar periféricos tales como el teclado, el mouse, la impresora, el escáner, etcétera, ahora una gran ventaja es que todos estos dispositivos pueden conectarse a la PC por medio de un mismo tipo de puerto, el USB [1].

La simplificación de la comunicación USB ofrece ventajas significativas debido a sus características, tales como:

- FÁCIL MANEJO. La configuración de la conexión resulta bastante sencilla para el usuario, en la mayoría de los casos ésta se realiza automáticamente por medio del sistema operativo, el cual carga los archivos de biblioteca y configuración necesarios para el correcto manejo del dispositivo, estos proveen a la PC de estructuras y protocolos de comunicación y dependerán del tipo de periférico que se conecte. En algunas ocasiones si el sistema operativo no cuenta con los elementos necesarios para la configuración del dispositivo, el usuario tendrá que agregarlos con ayuda de algún disco de instalación o bien por medio del acceso a la red de internet.

- RAPIDEZ. La tabla 2.1 muestra las diferentes velocidades de transmisión de datos. Estas dependen directamente del tipo de dispositivo que se conecte al puerto y pueden tomar tres valores diferentes:

TIPO	TASA Mbits/seg	VERSIÓN
Baja velocidad	1.5	1.0
Velocidad completa	12	1.1
Alta velocidad	480	2.0

Tabla 2.1. Velocidad de transmisión de datos vía USB.

  La primera se implementa para dispositivos de interfaz humana (HID) tales como ratones, teclados, etcétera, debido a que fueron las primeras aplicaciones que se le dieron al puerto USB, las otras dos se utilizan para conexión a internet y funciones o aplicaciones llevadas a cabo con versiones posteriores y que requieren la máxima velocidad posible. Los controladores incluidos en las PC fabricadas más recientemente posibilitan la comunicación con cualquiera de esas tres velocidades.

- CONFIABILIDAD. Los errores de comunicación ocurren en muy raras ocasiones. Su estructura de hardware permite aislar la transferencia de datos, de ruido e interferencia proveniente de fuentes externas debido a la malla metálica que rodea y aísla el par trenzado de conductores encaminados a dicha labor. Además su protocolo de

comunicación detecta y notifica sobre errores en la transferencia de datos de manera que estos puedan reenviarse, todo esto de forma automática y sin necesidad de que el usuario intervenga.

- VERSATILIDAD. Sus características en la comunicación lo hacen ideal para que diversos tipos de periféricos puedan usar esta conexión, aún cuando se trabaja con aplicaciones en tiempo real.

- BAJO COSTO. Tanto en tiempo de diseño y desarrollo, como para manufactura y comercialización.

- ALTA COMPATIBILIDAD. Ya que es soportado por varias plataformas y sistemas operativos, como son por supuesto Microsoft Windows, Linux y Mac OS, entre otros.

Por el tipo de estructura que posee, el USB tiene la capacidad de conectar una gran cantidad de dispositivos a la vez, pudiendo llegar a conectar hasta 127 en forma simultánea, permitiendo manejar tanto el tipo de comunicación síncrona como asíncrona.

Físicamente los datos se transmiten a través de un par trenzado de conductores "*D+*" y "*D-*", además el conector posee otra línea para alimentación *+5 V* y otra para *GND* tal que sirva de referencia para las señales de datos y alimentación. La figura 2.4 muestra la estructura básica del conector USB serie A y serie B. El conector serie A es el más implementado en diversos periféricos [1].

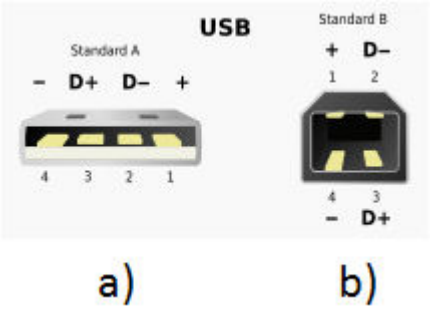


Figura 2.4. a) Conector USB estándar serie A. b) Conector USB estándar serie B.

2.3.1. LA COMUNICACIÓN VÍA USB.

La arquitectura externa de comunicación USB se basa en un dispositivo maestro denominado *HOST* el cual inicializa la comunicación, y uno o varios dispositivos o periféricos denominados *HUB* con los cuales realiza la misma. El protocolo de comunicación se divide en

dos partes, una para la detección y "*enumeración"* del dispositivo conectado, y otra para la aplicación propia del mismo según su propósito y su función.

La "*enumeración*" consiste en asignar un número índice a cada uno de los dispositivos que se conecten al *HOST* de forma que facilite el direccionamiento y flujo de datos durante la comunicación y de esta forma evitar el envío incorrecto de datos a otros dispositivos conectados. Durante este proceso, el software del dispositivo (*HUB*) responde a una serie de requerimientos técnicos solicitados por el *HOST* para aprender del mismo y poder comenzar con el intercambio de datos según su aplicación, estos requerimientos son llamados "*descriptores*" y son estructuras de datos que describen las capacidades y el modo de funcionamiento del dispositivo USB. El *HUB* debe identificar cada petición y responder enviando la información requerida para después tomar otra acción solicitada por el *HOST*.

En las PCs con sistema operativo Windows por ejemplo, éste se encarga de realizar el proceso de *enumeración* de forma automática. Cuando se conecta un nuevo dispositivo por el puerto USB, Windows debe localizar un archivo con extensión "*.inf*" que identifique el nombre y la localización de los driver del dispositivo. Si todos los archivos y requerimientos se encuentran bien configurados y funcionan correctamente, entonces el proceso de *enumeración* por lo general pasa desapercibido por el usuario.

Después de que el *HOST* ha intercambiado la información referente a la *enumeración* del dispositivo conectado y se han localizado y cargado los drivers necesarios para la configuración del mismo, el proceso de comunicación para su propósito específico puede comenzar.

Estas aplicaciones específicas de cada dispositivo pueden usar archivos de biblioteca o funciones estándar conocidas como "Windows API" o bien emplear algún otro software que permita realizar la transferencia de datos con el dispositivo. Estos protocolos de comunicación se fundamentan básicamente en el llamado "*token"* (también llamado "paso de testigo" en alguna literatura), en donde el *HOST* envía el "*token"* al *HUB* seleccionado (por medio del índice asignado durante la *enumeración*) para comenzar la transmisión de datos y éste le responde o regresa el *"token"* como respuesta, de esta manera se asegura que el dispositivo está listo para realizar la transferencia.

Por supuesto, la mayoría de los dispositivos requieren y manejan software adicional para el proceso de detección y corrección de errores durante la transmisión y otros eventos. Existen cuatro tipos de transmisión de datos para la comunicación asignada a un dispositivo USB: *asíncrona*, *de control*, *de interrupción* y *de carga* (*bulk*) cada uno con su formato y protocolo para cubrir las diferentes necesidades del mercado tecnológico actual [1]. Para el objetivo de este proyecto es innecesario indagar en cada uno de estos protocolos puesto que, como se verá

más adelante, se implementará un protocolo asíncrono simple que se asemeja al modelo de comunicación serie RS-232.

2.4. EL PUERTO BLUETOOTH.

El éxito del puerto de comunicación Bluetooth radica esencialmente en que el proceso de comunicación es de forma inalámbrica, lo cual representa su mayor ventaja sobre los demás puertos, adicionalmente brinda una gran facilidad en la conexión con otros dispositivos debido a sus protocolos, ya que ayudan la mayoría de las veces a que sea innecesario que el usuario configure dichas conexiones, lo que constituye otra gran ventaja con respecto a otros sistemas inalámbricos como podría ser por ejemplo el Wi-Fi [5].

El funcionamiento básico de la conexión es por medio de un enlace por radiofrecuencia que permite la transmisión de datos en la banda libre correspondiente a los 2.4 GHz empleando una modulación en frecuencia del tipo GFSK (*Gaussian Frecuency Shift Keying*). Puesto que existen artefactos que también utilizan enlaces en esta banda de radiofrecuencia, el protocolo Bluetooth realiza "*saltos*" en frecuencia usando 79 canales entre los 2.401 GHz y los 2.480 GHz con un ancho de banda de aproximadamente 1 MHz, esto sacrificando la eficiencia en el ancho de banda a cambio de confiabilidad, integridad y seguridad en el proceso de comunicación. Cada canal es divido en tiempos de 625 microsegundos en los cuales realiza un salto, llegando por lo tanto a realizar 1600 saltos en frecuencia por segundo [6].

Este tipo de conexión se realiza entre un dispositivo "*maestro*" y uno o varios dispositivos "*esclavos*", aunque la transmisión de datos solo se realiza de uno a la vez. La distancia máxima entre unos y otros es de 10 metros aproximadamente para transmisores de potencia de 1 mW o 0 dBm, y puede llegar a ser de hasta 100 metros para transmisores de potencia de 100 mW o 20 dBm.

Soporta dos tipos de enlace:

- Conexión asíncrona (ACL: Asynchronous Connectionless) orientada para transmisión de datos.
- Conexión síncrona (SCO: Synchronous Connection) orientada para transmisión de audio, voz y datos.

La conexión asíncrona puede llevarse a cabo de forma *asimétrica* o de forma *simétrica*. La *conexión asimétrica* ocurre cuando los dispositivos involucrados fungen uno como "maestro" y otro como "esclavo". La *conexión simétrica* se lleva a cabo cuando los dispositivos involucrados operan como "maestro" y "esclavo" a la vez.

La máxima velocidad en la transmisión de datos es de 1 Mbits/seg, sin embargo, para una conexión asimétrica del tipo ACL la velocidad de transmisión efectiva es de 721 Kbits/seg en una dirección y 56.7 Kbits/seg en dirección opuesta. Si la conexión es simétrica del tipo ACL entonces la velocidad de transmisión efectiva es de 432.6 Kbits/seg. Finalmente para conexiones del tipo SCO la velocidad de transmisión es de 64 Kbits/seg con un soporte de hasta tres canales por dispositivo y los cuales poseen un ancho de banda resguardado para realizar sin contratiempos la transmisión [5].

Los protocolos de comunicación Bluetooth pueden dividirse en 4 diferentes categorías, estas categorías y protocolos se muestran en la tabla 2.2 [6]:

CATEGORÍA	PROTOCOLOS
Núcleo de protocolos Bluetooth	Banda-base, LMP, L2CAP, SDP
Protocolo de remplazo de cable	RFCOMM
Protocolos de control de telefonía	TCS Binary, AT Commands
Protocolos adoptados	PPP, TCP/IP, OBEX, WAP, vCard, vCal, IrMC,WAE

Tabla 2.2. Protocolos Bluetooth.

La figura 2.5 muestra un diagrama de la estructura de la pila compuesta por los diferentes protocolos Bluetooth según la interacción que tienen unos con otros para el funcionamiento específico requerido [6].

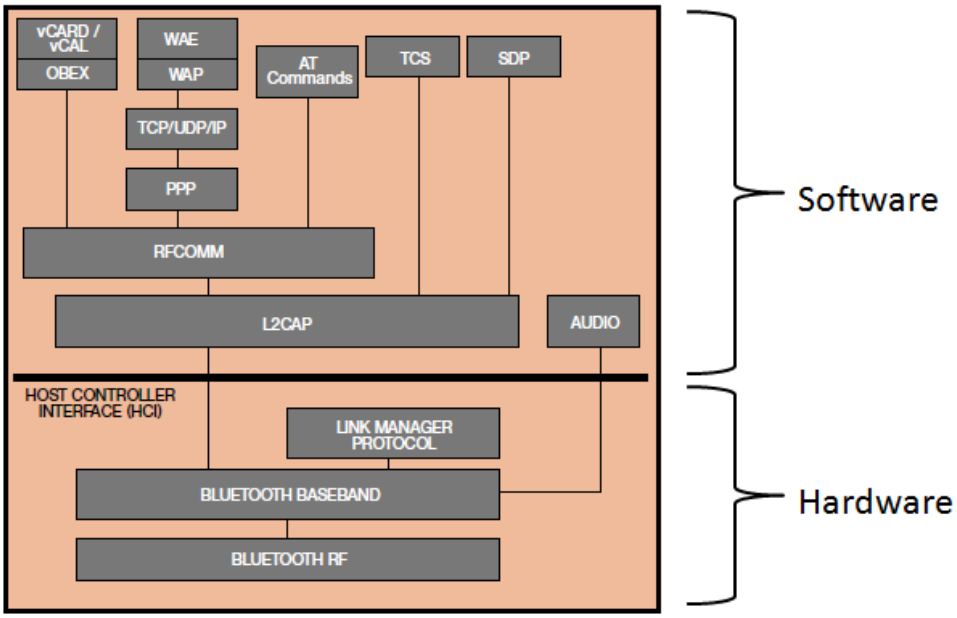


Figura 2.5. Pila de protocolos Bluetooth.

2.4.1. NÚCLEO DE PROTOCOLOS BLUETOOTH.

La ***Interface de Control del Huésped*** (HCI: *Host Controller Interface*) provee una interfaz de comando para el control de *banda-base*, el *Link Manager* y acceso al estado del hardware y los registros de control del dispositivo.

El ***protocolo de control Banda-Base*** habilita el enlace físico de RF entre dispositivos Bluetooth para formar una red para enlaces por Bluetooth, esta red es llamada "piconet".

El ***protocolo gestor de enlaces*** (LMP: *Link Manager Protocol*) es el responsable de la instalación y configuración del enlace entre unidades Bluetooth. Éste atiende a las cuestiones de seguridad como autentificación y cifrado mediante la generación, intercambio y control de enlaces y claves de cifrado. También trabaja con el control del tamaño de los paquetes de datos en la etapa Banda-Base.

El **protocolo *L2CAP*** (*Logical Link Control and Adaptation Protocol*) multiplexa los datos con una conexión del tipo *ACL* entre dos dispositivos. En caso de estar fungiendo como un dispositivo "maestro", le permite dirigir los datos al dispositivo "esclavo" apropiado por medio de identificadores de canal usados para reconocer cada terminal de conexión por separado. Asimismo, segmenta y vuelve a ensamblar los datos en paquetes que se ajusten a la máxima carga útil soportada por la HCI.

El ***protocolo de descubrimiento de servicios*** (*SDP: Service Discovery Protocol*) es la base para el descubrimiento de los servicios en todos los dispositivos Bluetooth. Esto es esencial para todos los equipos con este tipo de comunicación. Usando la información del dispositivo, los servicios y las características de los mismos se pueden consultar aún después de que una conexión entre dos o más dispositivos Bluetooth se establezcan. Otros protocolos de descubrimiento de servicios, tales como *Jini* o *UPnP* pueden ser utilizados junto con el protocolo Bluetooth *SDP*.

2.4.2. PROTOCOLO DE REMPLAZO DE CABLE.

El **protocolo *RFCOMM***, su nombre proviene del enlace por RF, orientado a emular los puertos seriales *COM* de una PC. Es una simple adaptación que permite emular la comunicación serie completa (de 9 pines) del puerto RS-232 a través de un canal establecido por el protocolo *L2CAP*, por ello tiene la habilidad de multiplexar y manejar diversos puertos serie virtuales dando a cada uno un registro de identificación de canal diferente, por lo que permite enlazar a varios dispositivos sin contraer problema alguno en la comunicación. El protocolo *RFCOMM* es vital para el proyecto presentado en esta tesis, puesto que es la base para el establecimiento de

la comunicación entre una PC y un teléfono móvil, el cual posee un modem integrado. El protocolo que nos permite interactuar con dicho modem se describirá más adelante.

2.4.3. PROTOCOLOS DE CONTROL DE TELEFONÍA.

El **protocolo *TCS Binary*** (*Telephony Control Specification Binary*) define el control de llamadas para datos y voz entre dispositivos Bluetooth. Está basado en el estándar de la Comisión Internacional de Telecomunicaciones (ITU) para el control de llamadas telefónicas Q.931 que provee comandos para notificaciones de llamada y establecimiento y finalización de la conexión de audio. Es utilizado ampliamente en perfiles de telefonía inalámbrica e intercomunicadores.

El **protocolo de *comandos AT*** (*"attention" commands*) se utiliza para interactuar con un modem y algunos dispositivos de audio. Dicho protocolo está dirigido para trabajar ya directamente en algún tipo de aplicación requerida. Tal como se muestra en la pila de protocolos de la figura 2.5 el envío y recepción de estos comandos está fundamentado en el establecimiento de una conexión de tipo puerto serie RS-232 por medio del protocolo *RFCOMM*, este puerto virtual a su vez es registrado y designado por el protocolo *L2CAP*. El manejo de comandos AT es una de las partes esenciales del proyecto presentado en esta tesis, ya que todo el establecimiento de la conexión telefónica del sistema de seguridad propuesto en este proyecto se lleva a cabo por medio de este protocolo.

2.4.4. PROTOCOLOS ADOPTADOS.

Los **protocolos *PPP*** (*Point to Point Protocol*) y ***TCP/IP*** (*Transmition Control Protocol / Internet Protocol*) son estándares de internet usados como base para la conexión tipo ***WAP*** (*Wireless Application Protocol*).

El **protocolo de intercambio de objetos *OBEX*** (*Object Exchange*) fue desarrollado por la IrDA (*Infrared Data Association*) para facilitar el intercambio de archivos entre dispositivos personales como PDAs y Laptops. Permite a los usuarios intercambiar datos, crear y eliminar carpetas u objetos, y especificar la carpeta de trabajo al dispositivo que funja como punto de acceso remoto. Las especificaciones Bluetooth han adoptado también los protocolos ***vCard*** orientado al intercambio de tarjetas de negocios digitales y ***vCal*** orientado al intercambio de calendarios.

Este tipo de protocolos "adoptados" carecen de interés para el proyecto presentado en esta tesis y no serán detallados más a fondo.

2.5. EL MODEM Y LOS COMANDOS AT.

Un modem es un dispositivo que se utiliza con el objetivo de modular y demodular señales eléctricas con el propósito de poder transmitir dichas señales a través de largas distancias de la manera más eficiente posible. Es importante mencionar que el modem como tal sólo adecua o prepara la señal para ser transmitida, pero no realiza la transmisión.

La modulación se realiza por medio de una señal sinusoidal de alta frecuencia llamada "portadora" a la cual se le modificará alguna o algunas de sus características (amplitud, frecuencia y fase) con base en las características de una señal proporcional a la información llamada "moduladora", de esta forma se obtendrá una señal de alta frecuencia que conlleva la información que se desea transmitir. Una vez realizada la transmisión, el sistema receptor, que también implementa un modem, deberá ser capaz de omitir la señal portadora (demodulación de la señal) para poder obtener dicha información.

Cuando un modem es configurado para establecer comunicación con un equipo de cómputo, el modem tiene que ser capaz de recibir la información y los datos de la computadora para posteriormente realizar la modulación y que puedan ser transmitidos, o bien, de forma inversa, el modem deberá realizar la demodulación de la señal recibida para posteriormente enviar los datos e información al equipo de cómputo. Es aquí en donde se hace evidente la importancia del entendimiento del estándar de comunicación serie RS-232, ya que éste es el protocolo que implementa el modem para lograr dicho objetivo.

Tomando como base la descripción hecha del puerto serie RS-232 anteriormente, el modem es un simple DCE que puede comunicarse con un DTE, tal como una computadora, por medio del puerto serie. Existen diversos tipos y marcas de modem y dependiendo del dispositivo seleccionado, la velocidad en la comunicación puede variar, al igual que las opciones de modulación y demodulación. Como se verá más adelante la tecnología actual permite inclusive que un teléfono celular pueda ser utilizado como modem si se configura correctamente su conexión con un sistema de cómputo.

2.5.1. TIPOS DE MODEM PARA UNA PC.

Los modem pueden clasificarse en "internos" o "externos" según como se deban instalar en una PC. Los modem internos están conformados por una tarjeta que debe ser instalada o conectada dentro de la PC, tienen la ventaja de no requerir una fuente de alimentación, ya que ésta la toman del propio equipo de cómputo. Los modem externos son mucho más sencillos de

conectar y desconectar puesto que utilizan alguno de los puertos RS-232 o USB de la computadora, los primeros modem que se desarrollaron ocupaban el puerto RS-232 y una fuente de alimentación externa para su funcionamiento, sin embargo, en la actualidad ya existen modem USB que también ocupan la alimentación de la PC.

Los modem también pueden ser clasificados según la técnica de modulación con la que trabajen, los primeros equipos utilizaban ***técnicas simples o básicas de modulación*** analógica como modulación en amplitud (*AM*), modulación en frecuencia (*FM*) y modulación en fase (*PM*). Ahora son muy implementadas las ***técnicas de modulación digital*** como *ASK (Amplitud Shift Keying)*, *FSK (Frecuency Shift Keying)* y *PSK (Phase Shift Keying)*, en las que éstas toman más de un bit para dicho proceso, es decir la modulación no se realiza de bit por bit, sino de un grupo de bits al mismo tiempo. Para esto es necesario codificar el grupo de 2 o más bits y obtener lo que se conoce como "símbolo" cuyo valor es único, ahora éste valor será el referente para el proceso de modulación. Los modem que utilizan esta técnica de codificación pueden realizar una velocidad de transmisión mayor.

La velocidad de transmisión de un modem está relacionada con su ***velocidad de modulación***, ésta se define como la cantidad de señales que son enviadas por unidad de tiempo y a esta unidad de medición se le conoce como "***Baudio***". Con las técnicas de modulación simples, el número de baudios coincide con el número de bits por segundo de la tasa de transmisión, mientras que con técnicas de modulación digital el número de *baudios* puede ser de la mitad o tercera parte de dicho número.

La relación que hay entre la velocidad de transmisión de datos en bits/seg y la velocidad de modulación en baudios está definida por la siguiente ecuación:

$$V_t = n \cdot V_m \tag{2.1}$$

donde "V_t" es la velocidad de transmisión en serie en bits/seg, "n" es el número de bits que conforman el *símbolo* que codifica el modem y "V_m" es la velocidad de modulación en baudios.

De la ecuación (2.1) se puede notar que la velocidad de modulación siempre será igual o menor que la tasa de bits por segundo de la transmisión, ya que "n" solo puede tomar valores enteros iguales o mayores que uno.

En la actualidad la mayoría de los modem trabajan con técnicas más avanzadas de modulación como es la ***modulación en cuadratura***:

- Modulación en cuadratura de amplitud (*QAM*).

- Modulación en cuadratura de fase (*QPM*).

- Modulación en cuadratura combinada de amplitud y de fase (*QAPM/AMPSK/QAMPSK*).

Esta última emplea un tipo de ***modulación combinada***, la cual consiste en implementar dos de las técnicas de modulación antes mencionadas al mismo tiempo. La más eficaz e implementada en los sistemas de comunicaciones actuales es la que introduce técnicas de modulación de amplitud con técnicas de modulación de fase.

El estándar *V29* por ejemplo, puede presentar hasta ocho fases diferentes (con 45° entre fases consecuentes) y dos amplitudes distintas por cada fase, de esta manera pueden representarse hasta dieciséis combinaciones diferentes con cuatro bits por baudio empleado en la modulación, por lo que consigue realizar una transmisión de datos a una tasa de 9600 bits/seg con una velocidad de modulación de 2400 baudios. La figura 2.6 muestra un diagrama vectorial, mejor conocido como *constelación*, de este tipo de modulación.

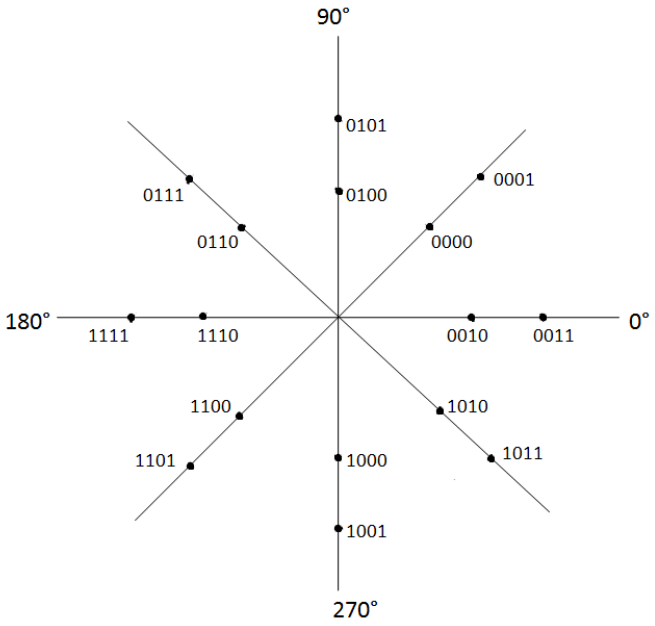


Figura 2.6. Constelación de la modulación en cuadratura de amplitud y fase V29 (*QAPM*).

Este tipo de modulación, empleada junto con diversas técnicas de codificación de datos y cancelación de eco que sirven para acrecentar la capacidad de transmisión de un modem, es el implementado actualmente. Hoy en día existen estándares que alcanzan tasas de transmisión de datos de 115,200 bits/seg y mayores.

2.5.2. COMANDOS AT.

En el año de 1981 una compañía de computación y comunicaciones llamada *Hayes Microcomputer Products, Inc.* desarrolló un modem "inteligente" capaz de recibir comandos codificados para su propia configuración o poder ejecutar ciertas operaciones, tales como marcar a un número telefónico, colgar o terminar una llamada en proceso, entre otras muchas acciones posibles mediante estos comandos. La compañía *Hayes* también desarrolló y registró a la par, un conjunto de comandos para interactuar con dicho modem. Este conjunto de comandos se conoce como "*conjunto de comandos Hayes"* o bien "*conjunto de comandos AT"*. Al paso del tiempo este conjunto de comandos se volvió tan popular que los desarrolladores comenzaron a copiar la estructura y funcionamiento básico de estos comandos para sus dispositivos modem, forzando a estandarizar de cierta forma los comandos para que pudieran ser aplicados con cualquier modelo y marca de modem que se desarrollara en un futuro. Sin embargo, en la realidad, cada compañía provee a sus comandos de algunas pequeñas diferencias que optimizan y caracterizan el funcionamiento de sus productos en particular. Prácticamente todos los comandos comienzan por una secuencia de dos letras, *"AT"* que resulta de una acotación de la palabra inglesa *"attention"*, de ahí que se les conozca como comandos AT [8].

Los comandos AT representan un protocolo de comunicación que nos permite la configuración y ejecución de ciertas operaciones realizadas por un modem. El modem no siempre se encuentra listo para recibir estos comandos o instrucciones, de ahí que prácticamente todos ellos sean precedidos por la cadena "*AT*" para que puedan operar, cuando el usuario envía esta cadena al modem le está indicando que debe estar atento para recibir algún tipo de comando a ejecutar. La tabla 2.3 muestra algunas cadenas empleadas en los comandos AT y su respectivo significado.

CADENA	SIGNIFICADO
AT	Atención (*Attention*)
ATA	Atención Responder (*Attention Answer*)
ATDT	Atención Marcado por tonos (*Attention Dial Tone*)
ATDP	Atención Marcado por Pulsos (*Attention Dial Pulse*)
AT+CHUP	Atención Comando Colgar (*Attention Command Hang Up*)
AT+BLDN	Atención Remarca al Último Número (Attention Begin Last Dial Number)

Tabla 2.3. Significado de algunas cadenas de comandos AT.

2.5.3. COMANDOS AT NOKIA PARA TELEFONÍA.

La compañía Nokia desarrolladora de equipos de telefonía móvil, provee a sus teléfonos celulares de la capacidad de ser utilizados como un modem, y como tal, la habilidad de responder a un conjunto particular de comandos AT. La compañía ha generado diversas recopilaciones de comandos a la par del desarrollo de sus productos, es por esta razón que dependiendo del modelo del teléfono celular, éste será o no capaz de responder a algunos de ellos, es decir, no todos los comandos AT de Nokia son aceptados por todos sus modelos telefónicos.

Para el desarrollo de este proyecto se utilizará un teléfono celular Nokia modelo 5610. Este modelo responde perfectamente a los siguientes comandos AT mostrados en la tabla 2.4 que son específicos para hacer y recibir llamadas:

COMANDO	ACCIÓN
ATA	El teléfono celular contestará una llamada entrante que esté en proceso.
ATDT XXX-XXXXX;	El teléfono realizará una marcación por tonos al número representado por la cadena "XXX-XXXXX" el ";" es para indicarle al modem que se llevará a cabo una llamada de voz y no de datos.
AT+CHUP	El teléfono finalizará la marcación o la llamada que se esté llevando a cabo.
AT+BLDN	El teléfono marcará al último número telefónico marcado con anterioridad.

Tabla 2.4. Comandos AT de Nokia para hacer y recibir llamadas.

Muchos conjuntos de comandos AT de Nokia se encuentran disponibles en la red y por supuesto se encuentran protegidos con derechos de autor, por lo que resulta un poco complicado encontrar las últimas versiones de los mismos. Sin embargo, los cuatro comandos descritos se mantienen constantes en todas las versiones anteriores y han dado resultados favorables en todas las pruebas realizadas con los mismos, empleando además diversos modelos de teléfonos Nokia en dichas pruebas [8].

2.6. RESUMEN Y SÍNTESIS.

En este capítulo se detallaron las características esenciales de los puertos RS-232, USB y Bluetooth de una PC así como sus protocolos de comunicación, e inclusive para el caso del puerto Bluetooth resaltando a aquellos que nos permiten emular el tipo de comunicación con un puerto serie RS-232 como son los protocolos *L2CAP* y *RFCOMM*. De esta manera será posible establecer la comunicación entre la PC y el teléfono celular vía Bluetooth.

El caso particular del puerto USB se retomará más adelante en los capítulos 2 y 5 cuando se haga uso de un archivo de biblioteca específico de *Microchip Technology Inc.* que permite emular la comunicación USB con un microcontrolador *PIC18F4550* como si fuera vía puerto serie RS-232, dicho microcontrolador será usado como interfaz entre los sensores, el audio del teléfono y el equipo de cómputo. En esos mismos capítulos se detallará la forma en la que el sistema de control, establecido en la PC por medio del ambiente Visual Basic, reconocerá y configurará los puertos USB y Bluetooth para poder llevar a cabo la comunicación con la interfaz del microcontrolador y el teléfono celular.

En este capítulo también se mencionaron algunos comandos AT que nos permitirán controlar los procesos de inicio y finalización de una llamada con el teléfono celular. En el próximo capítulo se detallará la forma en que es posible enviar dichos comandos a través de un puerto serie con ayuda de Visual Basic 6.0.

CAPÍTULO 3.
HERRAMIENTAS PARA PROGRAMACIÓN DEL HARDWARE.

Los microcontroladores y microprocesadores únicamente son capaces de entender datos y comandos en formato binario, a esto también se le conoce como *lenguaje de máquina*. Los equipos de cómputo no escapan a ello, puesto que su parte medular es un microprocesador. El *lenguaje de máquina* presenta alto grado de complejidad, puesto que para realizar una tarea relativamente simple se requiere de una elevada cantidad de instrucciones, que por ser únicamente combinaciones de unos y ceros no resultan amigables para el programador, por esta razón desde los inicios de la computación se han desarrollado lenguajes denominados de *alto-nivel* para facilitar la comunicación y programación de una computadora o un microcontrolador.

Los lenguajes de alto nivel están basados en el uso de nemónicos que sirven para identificar los componentes elementales del programa y los datos que se vayan a implementar en el mismo. Dependiendo del lenguaje de programación estos componentes se pueden llamar *rutinas*, *procedimientos* o bien *funciones*, anexo a esto, debe poseer una sintaxis específica que permita representar sin equivocación o confusión, las operaciones que se desean realizar.

En este capítulo se estudiarán las características y propiedades básicas de los lenguajes de programación *Microsoft Visual Basic 6.0* y *CCS PIC C Compiler,* los cuales serán empleados en la elaboración de una interfaz que sirva como enlace entre los sensores y periféricos del sistema de seguridad con el equipo de cómputo que nos permitirá la interacción con dicho sistema.

3.1. INTRODUCCIÓN A LOS LENGUAJES DE PROGRAMACIÓN.

Un programa de cómputo, se encuentra estructurado por variables y constantes que representan los datos con los que se requiere trabajar, y por algoritmos constituidos por funciones y/o procedimientos que representan las sentencias que operan sobre los datos adquiridos. Para que un microcontrolador o un equipo de cómputo sea capaz de entender un programa desarrollado en algún lenguaje de alto nivel, éste debe ser traducido a lenguaje de máquina, a este proceso se le denomina *Compilación* o bien *Interpretación* dependiendo del tipo de *implementación* que provea el lenguaje de programación utilizado.

La *implementación* de un lenguaje de programación se refiere a la forma de traducir y ejecutar un programa. El lenguaje C por ejemplo, es un lenguaje que implementa una *Compilación* puesto que genera un archivo o programa alterno, en lenguaje de máquina, conformado por la traducción del programa fuente, para que el sistema de esta manera pueda interpretarlo y consecuentemente ejecutarlo. El ambiente Visual Basic se encuentra en la rama de lenguajes que implementan una *Interpretación* puesto que en lugar de traducir todo el código y generar un programa alterno en lenguaje de máquina, éste se encarga de traducir solamente una rutina o instrucción conforme se va requiriendo y avanzando en la ejecución del programa.

Existen otras formas de diferenciar a los lenguajes de programación y los programas que en ellos se desarrollan. Éstos pueden clasificarse en *secuenciales*, *interactivos* y *orientados a eventos* [4].

Los programas *secuenciales* son aquellos que inician, leen los datos que necesitan, realizan las operaciones y los cálculos indicados según el algoritmo descrito en el programa y posteriormente imprimen o guardan en memoria los resultados obtenidos. La propiedad más importante de los programas secuenciales es que pueden ejecutarse (inclusive en ciclos infinitos) sin la necesidad de intervención humana. Muchos programas desarrollados en lenguaje C pueden pertenecer a este tipo de clasificación.

Los denominados *interactivos* son aquellos que requieren forzosamente la participación del usuario para la administración de los datos y la elección de las operaciones que se deseen realizar. *MATLAB* podría verse como un ejemplo de este tipo de programas.

Los programas *orientados a eventos* son los típicos de Windows como Excel, Power Point, etcétera. Cuando uno de éstos arranca, lo que sucede es que el programa espera a que el usuario realice ciertas acciones específicas, las cuales son llamadas *eventos*, para responder a ellas según lo programado para tal suceso en especial. De esta manera, dichos programas funcionan con base en un conjunto de eventos predeterminados cuyas respuestas están conformadas por algoritmos, operaciones, datos y variables que los caracterizan.

3.2. MICROSOFT VISUAL BASIC 6.0.

El ambiente Visual Basic 6.0 es un lenguaje de programación visual, es decir, se encuentra diseñado para trabajar en el entorno gráfico del sistema operativo Windows, esto nos permite crear interfaces que estén dotadas de ventanas, menús y controles con los que estamos familiarizados en dicho sistema operativo, por lo que un gran número de tareas pueden ser ejecutadas por medio de instrucciones o sucesos comandados por el mouse de la computadora. Además, por su forma de interacción y ejecución, se clasifica como un lenguaje de programación orientada a eventos [4].

3.2.1. CARACTERÍSTICAS DEL LENGUAJE VISUAL BASIC.

La mayoría de los elementos gráficos que forman parte de una aplicación para el sistema operativo Windows se encuentran disponibles en el ambiente Visual Basic como un tipo de *control*. Algunos de los controles más comunes e implementados son:

- Botones de comando (*Command Button*).
- Botones de opción (*Option Button*).
- Botones de selección (*Check Box*).
- Cajas de texto (*Text Box*).
- Cajas de gráficos (*Picture Box* e *Image*).
- Cajas de selección (*List Box* y *Combo Box*).
- Barras de desplazamiento horizontal y vertical (*HScroll Bar* y *VScroll Bar*).

Aunque por supuesto existen muchos otros de gran utilidad y más adelante se describirán de forma breve los de mayor interés para este proyecto.

Todo control posee un nombre específico que lo caracteriza en el momento de hacer referencia a él durante la ejecución del programa, de esta forma se evita la posible confusión que puede presentarse cuando se utiliza un gran número de controles del mismo tipo. El lenguaje de programación asigna automáticamente un nombre a cada uno cuando el programador decide implementarlos, pero es posible cambiarlos o personalizarlos para que se facilite la identificación por parte del desarrollador.

La gran mayoría de las aplicaciones en Windows están formadas por una o varias ventanas, éstas fungen como contenedores para los controles con los cuales el usuario interactúa. Una ventana en Visual Basic posee el nombre de formulario (*form*) y al igual que los controles su nombre puede ser modificado para comodidad del programador.

Visual Basic 6.0 es un lenguaje basado en objetos, donde todos los formularios y tipos de controles representan entidades genéricas que componen en su conjunto al programa o la aplicación. De esta manera al tipo de control se le llama *clase* y cada control o formulario representa un *objeto* de esa clase.

Todo objeto del programa, sea formulario o control, posee un conjunto de *propiedades* que caracterizan y conforman su aspecto gráfico y la forma en que responderán a ciertos eventos o acciones por parte del usuario. El número de propiedades disponibles puede variar para cada tipo de objeto, y el nombre de cada propiedad se encuentra definido por el lenguaje de programación y no puede ser modificado por el desarrollador. Todas las propiedades de un objeto poseen valores propios (lógicos o alfanuméricos) predeterminados al momento de su implementación, y éstos pueden ser modificadas en tiempo de diseño y casi siempre también en tiempo de ejecución.

La nomenclatura requerida para acceder a alguna propiedad de un objeto se define escribiendo primero el nombre específico del objeto seguido de un punto y luego el nombre de la propiedad que nos interesa modificar o caracterizar. Por ejemplo:

Linea1.Color = "Rojo"

Donde *Linea1* es el nombre del objeto, *Color* es el nombre definido para la propiedad y la cadena *Rojo*, encerrada entre comillas, es una variable que representa el valor que caracteriza al color rojo dentro del lenguaje de programación.

Como ya se mencionó con anterioridad, Visual Basic 6.0 es un lenguaje orientado a eventos, estos *eventos* son representados por las acciones realizadas por las interfaces de comunicación de la PC (periféricos como teclado, mouse y otros) que pueden estar comandados por el usuario o por algún sistema externo al equipo de cómputo. De esta manera, cuando un evento se lleva a cabo sobre un cierto tipo de control, éste deberá responder con una determinada función o procedimiento que realice la acción programada por el desarrollador para ese evento en particular. Por esta razón durante el diseño de la aplicación es necesario que el programador especifique cada uno de los eventos por los cuales todos los controles empleados en dicha aplicación responderán con sus procedimientos y métodos. Puede ser inclusive que un solo control responda de diferente forma, para diversos tipos de eventos que sobre él se lleven a cabo.

Los *métodos* son funciones pre-programadas por el lenguaje para realizar operaciones y funciones de uso muy común en diversas aplicaciones. Cada tipo de objeto o control posee sus propios métodos y facilitan al desarrollador el realizar tareas típicas y comunes sin la necesidad de ser programadas [4].

3.2.2. DESCRIPCIÓN DE CONTROLES, PROPIEDADES Y EVENTOS DE INTERÉS.

Una de las características que hacen de Visual Basic 6.0 muy popular entre los programadores, sobre todo novatos, es la facilidad y la rapidez con que se puede aprender a manejar dicho lenguaje.

A continuación se describirán brevemente los controles que serán empleados para la realización del proyecto, haciendo mención además de algunas de sus propiedades. También se detallarán los posibles eventos que permiten la interacción con el sistema que se propone en este documento. En el capítulo 5 se detallará más a fondo la configuración de los controles más vitales para el proyecto junto con el código que represente las funciones y procedimientos que respondan a cada uno de los eventos de interés para el sistema, mientras que en el capítulo 4 se describirán de forma más general los algoritmos que representen dichas funciones.

3.2.2.1. CREACIÓN DE UN NUEVO PROYECTO EN VISUAL BASIC 6.0.

Cuando se empieza a desarrollar una nueva aplicación en Visual Basic 6.0 el lenguaje lo reconoce como un nuevo *proyecto*. Un programa se encuentra formado básicamente por los siguientes componentes: los *formularios* y los *módulos*. Cuando se crea un nuevo proyecto se generan una serie de ficheros (al menos dos) que contienen la información en conjunto de todo el programa, es decir, qué formularios y módulos serán empleados en el proyecto. También se genera un fichero por cada formulario o módulo que se diseñe, y éste contendrá la información particular de cada elemento. Es importante mencionar que todos los ficheros deben guardarse en el mismo directorio y con el nombre adecuado para que el sistema funcione adecuadamente.

Para crear un nuevo proyecto solo es necesario ingresar a Visual Basic 6.0 dar clic en el menú *Archivo* y luego en la opción *Nuevo Proyecto*, emergerá una nueva ventana que presenta diversas opciones y tipos de aplicación a desarrollar, se elige la opción *EXE estándar* y luego se da clic en el botón *Aceptar* para comenzar a trabajar en un programa de ejecución común, que son con los que estamos acostumbrados a trabajar en Windows (extensión *.exe*). La figura 3.1 muestra una imagen del menú desplegado para crear un nuevo proyecto.

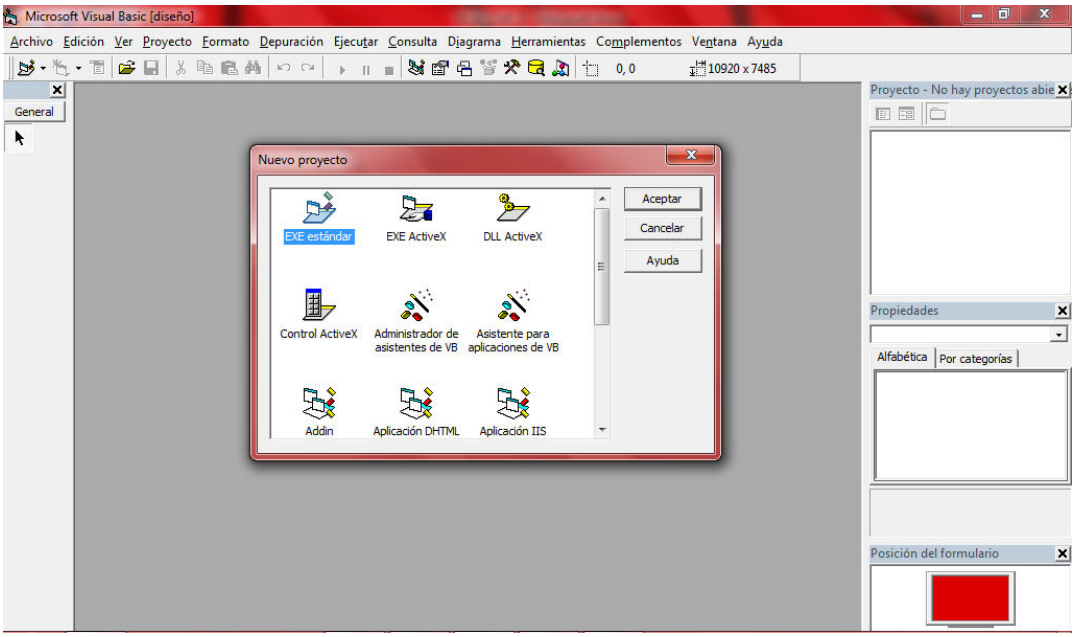


Figura 3.1. Menú para generar un nuevo proyecto en Microsoft Visual Basic 6.0.

Inmediatamente después de haber terminado el proceso descrito anteriormente emergerá el formulario base para la aplicación a desarrollar y aparecerá el menú con los diversos controles y las propiedades básicas de los objetos seleccionados. Puede verse la figura 3.2 para comprender mejor lo descrito anteriormente.

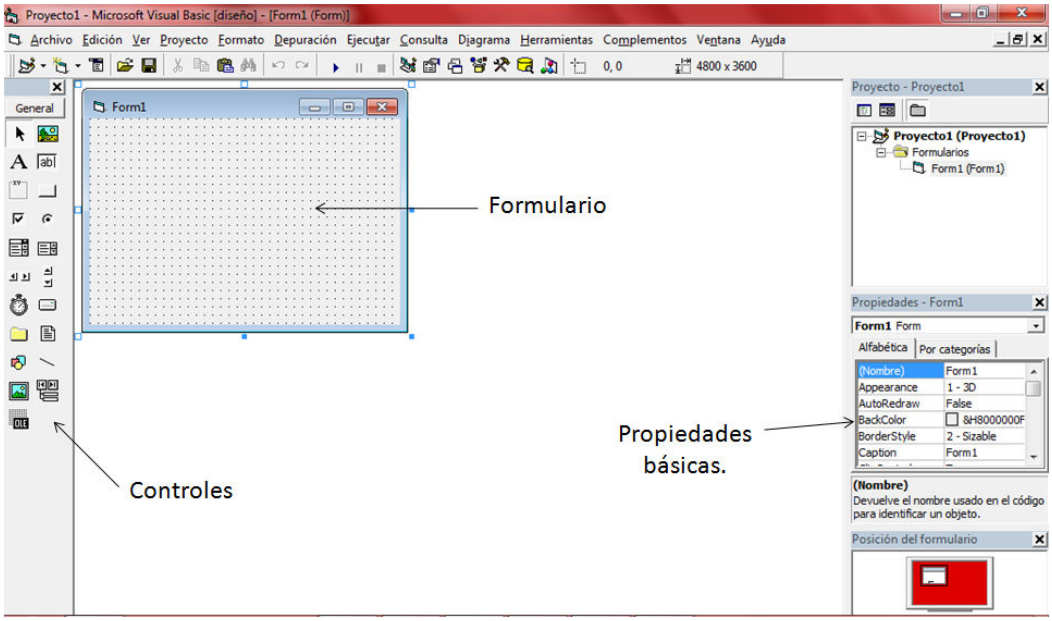


Figura 3.2. Pantalla de trabajo que se muestra al iniciar un nuevo proyecto en Visual Basic 6.0.

Una vez llegado a este punto podremos modificar el tamaño del formulario, agregarle objetos, tales como botones o imágenes, cambiar su nombre o título, color, etcétera. La lista de

propiedades básicas dependerá del objeto seleccionado, ya sea el propio formulario o algún botón, gráfico, caja de texto o cualquier otro control incluido dentro del mismo.

3.2.2.2. MANEJO DE FORMULARIOS (Form).

Los *formularios* son las zonas de la pantalla sobre las que se diseña el programa y sobre las que se sitúan los controles o herramientas que se implementarán. Al ejecutar el programa, el *formulario* se convertirá en la ventana de la aplicación, donde aparecerán los botones, el texto, los gráficos, etcétera.

Exteriormente, los formularios tienen una estructura similar a la de cualquier ventana. Sin embargo, también poseen un código de programación que estará escrito en *Basic*, y que controlará algunos aspectos del formulario, sobre todo en la forma de reaccionar ante las acciones del usuario y los eventos de interés. El formulario y los controles en él situados serán el esqueleto o la base del programa. Una aplicación puede tener varios formularios, pero siempre habrá uno con el que arrancará la aplicación.

Cuando se arranca una aplicación, o más en concreto cuando se visualiza por primera vez un formulario se producen varios eventos consecutivos: ***Initialize***, ***Load***, ***Activate*** y ***Paint***. Al ocultar, cerrar o eliminar un formulario se producen otra serie de eventos: ***Deactivate***, ***QueryUnload***, ***Unload*** y ***Terminate***. Cada uno de estos eventos se puede aprovechar para realizar ciertas operaciones por medio de la función correspondiente [4].

Además de estos eventos que se producen durante la carga y descarga de los formularios, es importante mencionar algunas propiedades básicas de los mismos. Los formularios al igual que algunas otras herramientas o elementos de una aplicación en Visual Basic, poseen la propiedad *Caption* la cual representará al título visual del elemento. Para un formulario el nombre asignado por *default* es el de *FormX* donde *X* representa algún número entero comenzando con *1* para el primer formulario de la aplicación. La propiedad *Caption* de un formulario o del resto de los elementos que también poseen dicha propiedad puede ser modificada tanto en tiempo de diseño como en tiempo de ejecución.

Para modificarla en tiempo de diseño solo es necesario señalar el formulario con un clic y acudir al *menú de propiedades* para modificar esta y otras propiedades básicas de los elementos seleccionados. La figura 3.2 ya descrita anteriormente, señala el área de interés para modificar todas las propiedades básicas. Cuando modificamos alguna propiedad en tiempo de diseño estamos estableciendo valores de inicio a nuestro sistema inclusive antes de que se cargue el primer formulario del programa, de esta manera el programa tomará estos valores y datos como los preestablecidos por el sistema.

Para modificar cualquier propiedad en tiempo de ejecución sólo es necesario seguir la nomenclatura ya descrita y enlazarla con algún evento de interés para nosotros como lo puede

ser la propia carga del formulario. Con la figura 3.3 se puede observar un ejemplo del código necesario para cambiar la propiedad *Caption* en tiempo de ejecución. Es importante visualizar las dos listas desplegables ubicadas en la parte posterior del área de código. Para ingresar a la ventana de edición de código solo es necesario dar un doble clic en el elemento con el que nos interese trabajar, llámese formulario, botón de comando o cualquier otro que haya sido anexado dentro del proyecto a desarrollar (en este ejemplo al formulario). La lista de selección del lado izquierdo señala el elemento del sistema que trabajará con el código que se vaya a ingresar, mientras que la lista del lado derecho señala el evento que provocará que dicho código sea ejecutado.

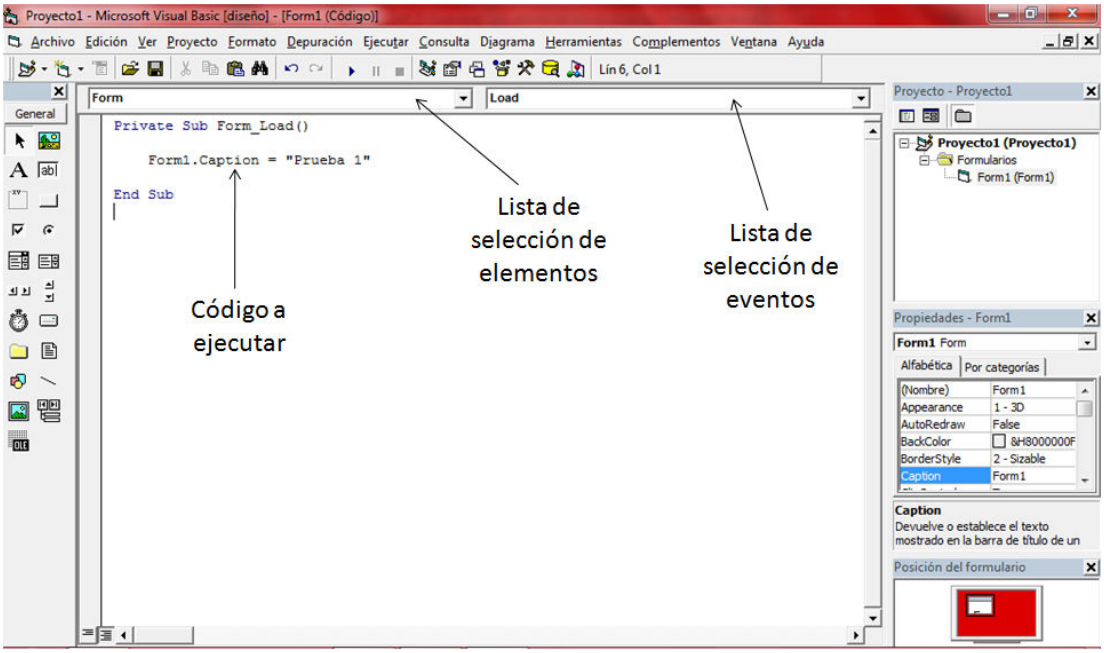


Figura 3.3. Ventana para editar código en Visual Basic 6.0.

Es posible cambiar o editar código de otro elemento del sistema simplemente desplegando la lista de selección de elementos y eligiendo cualquiera de los que ahí se muestren. De igual manera será posible editar código para uno o varios eventos disponibles para el elemento seleccionado, los cuales serán mostrados en la lista de selección de eventos y cuyo número y tipo dependerán del elemento con el que se esté trabajando. La figura 3.3 ejemplifica el proceso de cambio de la propiedad *Caption* del formulario en tiempo de ejecución cuando se efectúa el evento *Load* del *Form1*, si se prueba dicho código dando clic en el botón de *Iniciar* se podrá observar que desde el momento en que el formulario salta a la vista, el título del mismo ha cambiado de *Form1* a *Prueba 1* tal como se muestra en la figura 3.4. Es importante mencionar que es mucho más sencillo modificar propiedades tan básicas como *Caption* en tiempo de diseño que en tiempo de ejecución, ya que rara vez es necesaria dicha modificación, pero para un posible principiante puede ser de mucha ayuda el "jugar" a modificar estas propiedades de ambas formas, para familiarizarse con el entorno de programación.

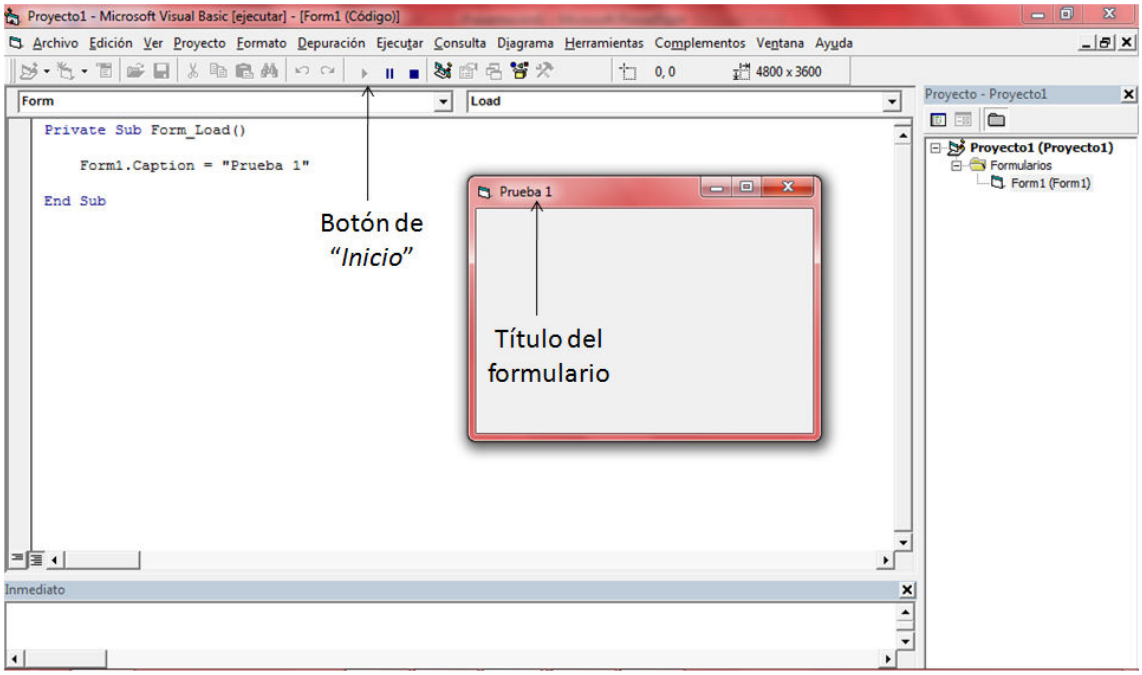


Figura 3.4. Ejecución de un programa de prueba para modificar la propiedad "*Caption*" de un formulario.

A continuación se describirán algunas propiedades importantes de los formularios:

- ***Caption***. Representa el título de la ventana o formulario.
- ***BorderStyle***. Se refiere al tipo de ventana que representa el formulario, con esta propiedad se puede controlar si la ventana puede minimizarse, maximizarse o inclusive si es posible modificar su tamaño con ayuda del ratón. Tiene 6 posibles valores: *0 – None*, *1 – Fixed Single*, *2 – Sizable*, *3 – Fixed Dialog*, *4 – Fixed ToolWindow* y *5 – Sizable ToolWindow*.
- ***Enabled***. Habilita o deshabilita el formulario, está propiedad tiene un valor de tipo booleano, es decir, sólo puede tomar el valor de *verdadero* o *falso*. Cuando tiene el valor *falso* deshabilita por completo el formulario, haciendo imposible el trabajar con ningún elemento o propiedad que tenga enlace con el mismo.
- ***Visible***. Esta propiedad también posee un valor de tipo booleano y permite que el usuario tenga o no visibilidad del elemento, en este caso del formulario.

Existen muchas otras propiedades para dar formato y configurar los formularios, el dominio de todos ellos radica esencialmente en la experimentación y la práctica del programador.

3.2.2.3. EL BOTÓN DE COMANDO (Command Button).

Las propiedades básicas del *botón de comando* son ***Caption***, ***Enabled*** y ***Visible***, éstas tienen el mismo objetivo que en el caso del formulario y muchos otros controles disponibles. De igual manera pueden ser editados tanto en tiempo de diseño como en tiempo de ejecución.

El evento predeterminado para que un *botón de comando* ejecute su código correspondiente es el evento ***Click***, éste se lleva a cabo cuando el usuario hace clic sobre el *botón de comando*, y es el que será implementado para la interacción entre éste y la interfaz gráfica preparada para este proyecto, aunque por supuesto existen muchos otros eventos a los que puede responder dicho elemento. La figura 3.5 muestra la forma visual de un botón de comando, para incluir este tipo de control en una aplicación basta con seleccionarlo desde la *barra de controles* (véase figura 3.2) y dibujarlo dentro del área del formulario de interés para el programador.

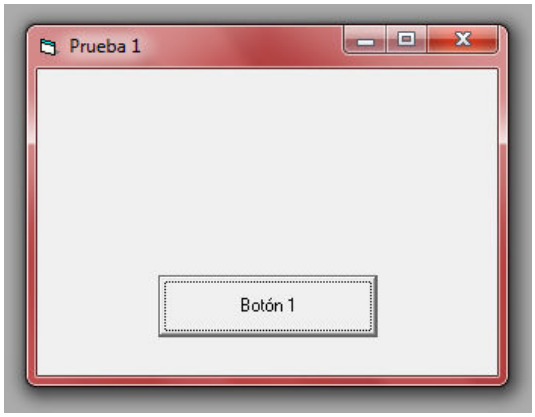


Figura 3.5. Formulario estándar con un *botón de comando* incluido.

3.2.2.4. LA CAJA DE TEXTO (Text Box).

Las *cajas de texto* son uno de los principales controles dentro de Visual Basic, ya que nos permiten interactuar con la entrada y salida de datos de cualquier tipo, ya sean numéricos o bien cadenas de caracteres. La propiedad más importante de este control es ***Text***, la cual representa aquello que esté contenido dentro la propia *caja de texto*. También posee las propiedades ***Enable*** y ***Locke***, ésta última cuando es establecida en *True* hace que la caja de texto sea de sólo lectura.

Los eventos que se pueden programar para este control rara vez son utilizados, quizás el más implementado sea el evento ***Change*** que se lleva a cabo cuando el usuario realice una acción que modifique el contenido de la caja de texto. También se pueden programar para los eventos ***Click*** y ***KeyPress***. La figura 3.6 muestra un formulario en ejecución que muestra la forma visual de una caja de texto en su interior.

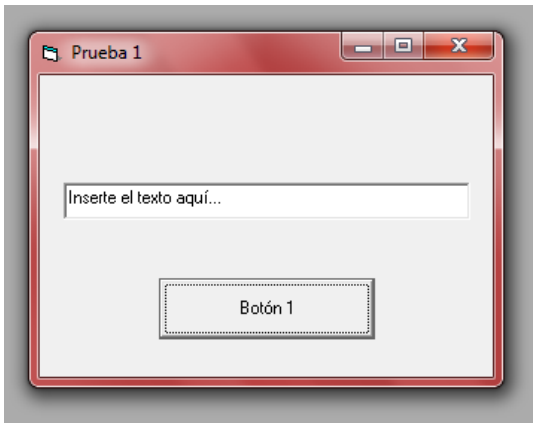


Figura 3.6. Formulario estándar con un botón de comando y una caja de texto en su interior.

3.2.2.5. EL CONTROL DE TIEMPO (Timer).

Este control sirve para programar que una o varias acciones se lleven a cabo con cierta periodicidad, esto se puede lograr produciendo eventos de forma automática cada cierto número de milisegundos. Es vital para realizar aplicaciones con movimiento de objetos o monitorear e interactuar con el estado de otros controles del programa.

Las propiedades más importantes de este control son ***Enabled*** que sirve para activar o desactivar el mismo y por ende todo lo que con él se esté llevando a cabo, y la propiedad ***Interval*** que sirve para establecer el periodo de trabajo del *Timer* en milisegundos. La figura de este control sólo es visible al programador en tiempo de diseño, durante la ejecución es invisible para el usuario.

Figura 3.7. Forma visual del control *Timer* en tiempo de diseño.

3.2.2.5. EL CONTROL DE COMUNICACIÓN SERIE (MSComm).

El control *MSComm* no se encuentra disponible de forma predeterminada en la barra de controles de Visual Basic, por lo que es necesario agregarlo. Para llevar a cabo esto sólo es necesario dar clic con el botón secundario del ratón en la barra de controles y luego acceder o dar clic en el menú "*Componentes…*" en el que aparecerán una gran cantidad de controles extra que pueden ser utilizados para realizar aplicaciones en Visual Basic 6.0. La figura 3.8 muestra una imagen de dicho menú en donde ya se ha seleccionado la casilla correspondiente al control

Microsoft Comm 6.0, una vez realizada tal acción solo basta con dar clic en el botón aceptar para poder acceder a dicho control directamente de la barra de controles en la pantalla de Visual Basic.

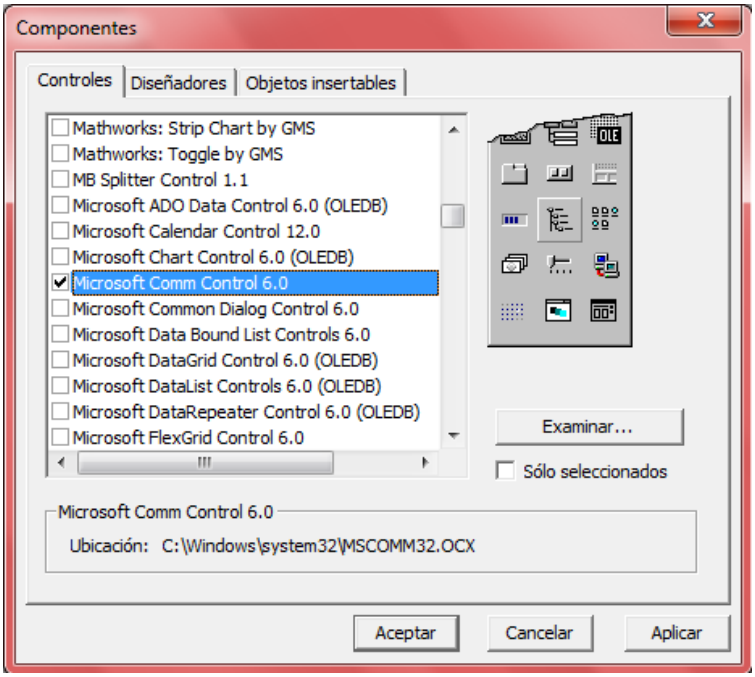


Figura 3.8. Menú de controles extra disponibles para Visual Basic 6.0.

El control MSComm sirve para enviar y recibir datos a través de un puerto de comunicación serie del tipo RS-232 utilizando los métodos *Output* e *Input*. Tiene muchas propiedades importantes que sirven para la propia configuración del puerto, algunas de las más importantes son:

- ***CommPort***: Su valor es un entero que indica el número de puerto con el que se va a trabajar. Ej. 1,2,3,…,etc.
- ***Settings***: Su valor es una cadena números divididos en cuatro áreas que indican primero la tasa de bits de transferencia por segundo, luego si se hará o no uso de bits de paridad, seguido de el número de bits de datos a manejar durante la transferencia y por último el número de bits de parada (*stop bits*). Ej. “9600,N,8,1” (N: Sin bits de paridad).
- ***InputMode***: Puede tomar dos valores distintos 0 – InputModeText ó 1 – InputModeBinary. Tal como se detalló en el capítulo anterior, la transmisión serie a través del puerto RS-232 puede llevarse a cabo en binario ó como texto en código ASCII.
- ***PortOpen***: Cuando está propiedad es puesta en *True* el programa ejecuta los pasos necesarios para abrir el puerto y verifica la conexión con el dispositivo conectado a él,

en caso de que la conexión esté mal configurada o el dispositivo no se encuentre bien conectado entonces el sistema marcará un error en la conexión. Cuando la propiedad es puesta al valor *False* el sistema cierra el puerto y termina la comunicación.

Un ejemplo del código necesario para configurar dicho control y establecer comunicación con un dispositivo, por ejemplo un modem, sería de la forma:

```
MSComm1.CommPort = 4                ' Se establece el número del puerto serie a utilizar.
MSComm1.Settings = "9600,N,8,1"     ' Se configuran las características de comunicación
                                    ' con el dispositivo.
MSComm1.InputMode = InputModeText   ' Los datos se transmitirán como texto en código ASCII.
MSComm1.PortOpen = True             ' Se abre el puerto indicado y se establece
                                    ' la comunicación con el dispositivo.
```

Una vez establecida la comunicación con el dispositivo se puede proceder a enviar datos o comandos por medio de los métodos antes mencionados, por ejemplo:

```
MSComm1.Output = "ATDT 555-1234567;"  ' El dispositivo, en este caso un modem, recibe el comando AT
                                      ' de marcado por tonos al número 555-1234567.
Variable = MSComm1.Input              ' Guardamos los datos enviados
                                      ' por el dispositivo en una variable.
```

La figura 3.9 muestra la imagen del control *MSComm* para ser implementado en una aplicación en Visual Basic, al igual que el control *Timer*, éste es invisible para el usuario en tiempo de ejecución. En capítulos posteriores se retomará el trato con este control dado que es vital para este proyecto de tesis en los procesos de comunicación entre los elementos del sistema de seguridad.

Figura 3.9. Imagen del control MSComm disponible en Visual Basic 6.0.

Los dispositivos con los cuales se desee trabajar posean protocolos de comunicación y archivos de biblioteca o *drivers* que permitan emular un puerto RS-232, ya que Visual Basic 6.0 puede reconocer dichos puertos y por consiguiente se podrá interactuar con dichos dispositivos mediante alguna aplicación en este lenguaje. En el capítulo anterior se detalló la estructura de la pila de protocolos de comunicación del puerto Bluetooth, se describieron los protocolos L2CAP y RFCOMM que permiten emular la comunicación serie tipo RS-232. Más adelante en este mismo capítulo se describirá el archivo de configuración que permite realizar la misma labor con un microcontrolador PIC a través de su puerto USB.

3.3. CCS PIC C COMPILER (LENGUAJE C PARA PIC).

Existe una rama de compiladores conocidos como *cross-compiler*, éstos funcionan en un procesador diferente al procesador objeto, el compilador PIC C de *Custom Computer Systems, Inc.* es de este tipo, puesto que el programa se edita y compila con ayuda de una PC para que el código resultante en lenguaje de máquina sea instalado en alguno de los microcontroladores PIC de *Microchip, Inc*. Es importante mencionar que este compilador ha sido desarrollado para trabajar con los dispositivos de dicha compañía específicamente, por lo que suministra controladores para los mismos.

El compilador PIC C utiliza el lenguaje C estándar para llevar a cabo los programas a diseñar, y suministra las directivas clásicas de dicho lenguaje además de otras particulares para la configuración y manejo de los microcontroladores PIC.

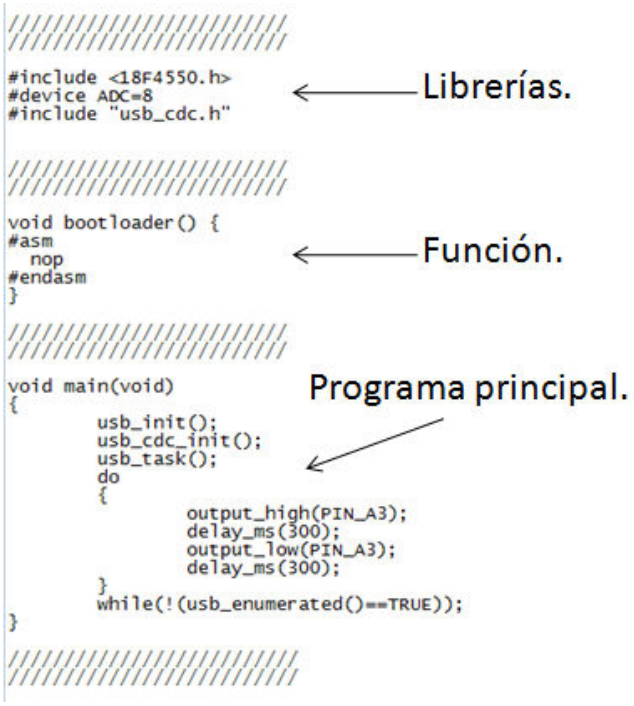
3.3.1. ESTRUCTURA DE UN PROGRAMA EN C DE CCS.

La estructura de un programa en el lenguaje C de CCS es prácticamente la misma que la implementa en una versión de lenguaje C estándar. Se comienza declarando las directivas de pre-procesado que se utilizarán en el programa para manejo de funciones o métodos preestablecidos por el lenguaje, un ejemplo de ello podrían ser las funciones que permiten realizar operaciones matemáticas. Después se efectúa la edición del código del programa principal *main()* y las funciones a las que se hará llamado dentro del mismo, este código está conformado por un conjunto de instrucciones que indican cómo debe comportarse en todo momento el dispositivo, o en este caso particular el microcontrolador PIC. Al igual que muchos otros lenguajes de programación, también permite la edición de comentarios a la par del código, con el propósito de facilitar el proceso de desarrollo hacia el programador o programadores involucrados. La figura 3.10 muestra la estructura básica de un programa editado en PIC C de la compañía CCS.

3.3.2. ARCHIVOS DE BIBLIOTECA Y COMANDOS RELEVANTES DE PIC C.

Todos los microcontroladores poseen puertos de entrada y salida, algunos puertos muchas veces están más enfocados en señales de control y adquisición de datos y algunos otros para ser empleados en establecer comunicación con otros dispositivos. Este enfoque desarrollado para cada puerto depende enteramente de sus características, es decir, de los bloques funcionales que posean tales como: Convertidores A/D, Salidas PWM, protocolos de envío y recepción de datos como USART o I^2C.

Para hacer uso de estos bloques es necesario declarar su implementación desde un principio y hacer un llamado a los archivos de biblioteca correspondientes. En la mayoría de los casos, estos bloques vienen configurados para hacer uso de ellos con sólo llamar a sus funciones o métodos, pero es posible también configurar ciertas características, esenciales tal vez para alguna aplicación, de forma previa mediante otra serie de comandos extras. A continuación se hará una descripción breve de los archivos de configuración necesarios para implementar alguno de los bloques antes mencionados además de los comandos o instrucciones que permiten hacer uso de ellos.




```
////////////////////////////
////////////////////////////
#include <18F4550.h>
#device ADC=8
#include "usb_cdc.h"

////////////////////////////
////////////////////////////
void bootloader() {
#asm
  nop
#endasm
}

////////////////////////////
////////////////////////////
void main(void)
{
        usb_init();
        usb_cdc_init();
        usb_task();
        do
        {
                output_high(PIN_A3);
                delay_ms(300);
                output_low(PIN_A3);
                delay_ms(300);
        }
        while(!(usb_enumerated()==TRUE));
}
////////////////////////////
////////////////////////////
```

Figura 3.10. Estructura básica de un programa en C de CCS.

3.3.2.1. LECTURA Y ESCRITURA DE PUERTOS.

Puesto que las funciones de lectura y escritura de puertos en forma digital son las establecidas de forma predeterminada en todos los PIC no es necesario llamar a algún archivo de biblioteca para llevar a cabo dicha función. Las siguientes instrucciones muestran una de las formas de leer y escribir sobre los puertos de un PIC de forma digital:

```
// Declaramos una variable donde guardaremos el valor de entrada del puerto D.
Int1 dato1;              // El tipo de dato "int1" es de 1 solo bit.
char dato2;              // El tipo de dato "char" es no signado y de 8 bits.
dato1 = input(PIN_A1);   // Lee el pin 1 del puerto A y asigna su valor a la variable "dato1";
dato2 = input_D();       // Lee el puerto D y asigna su valor a la variable "dato2".
output_low(PIN_C0);      // Pone en valor "0" el pin 0 del puerto C.
```

```
output_high(PIN_C1);        // Pone en valor "1" el pin 1 del puerto C.
output_bit(PIN_C2, dato1);  // Escribe el valor de "dato1" en el pin 2 del puerto C.
output_B(dato2);            // Escribe el valor de "dato2" en el puerto B.
```

3.3.2.2. LECTURA DEL CONVERTIDOR A/D.

Para poder dar de alta el bloque del convertidor analógico – digital del microcontrolador, es necesario hacer uso de uno de los archivos de biblioteca del compilador:

```
#device ADC=8;  // Damos de alta el bloque de conversión A/D con 8 bits para el valor de retorno del dato.
```

Ahora para configurar cuál puerto y qué terminal se establecerá para realizar dicha función, dentro del programa principal se introducen las siguientes instrucciones:

```
setup_adc(ADC_CLOCK_INTERNAL);  // Se habilita el puerto y se especifica que trabajará con la señal de
                                // reloj del microcontrolador.
setup_adc_ports(ALL_ANALOG);    // Se configuran todos los pines que posean este bloque funcional para
                                // entradas analógicas.
set_adc_channel(0);             // Se da de alta el canal 0 del convertidor.
delay_us(10);                   // Se da un retardo de 10 microsegundos
                                // para que se estabilice la configuración.
```

Finalmente la instrucción que permite leer el valor establecido a través del convertidor A/D es la siguiente:

```
char dato;          // Se declara la variable "dato" donde guardaremos la lectura.
dato = read_adc();  // Se lee el valor del puerto A/D y se asigna a la variable dato.
```

3.3.2.3. COMUNICACIÓN USB-CDC.

El microcontrolador *PIC18F4550* de *Microchip* posee un bloque funcional que lo habilita para establecer comunicación a través del puerto USB con otro dispositivo, por ejemplo una PC. Tal como se detalló en el capítulo anterior, el llevar a cabo esta labor comprende la realización de varios procesos de configuración previos antes de establecer propiamente la comunicación USB. Entre los archivos de biblioteca del compilador existen algunos específicos para lograr dicho objetivo de manera sencilla, además de existir un archivo de configuración y un conjunto de instrucciones especiales para lograr que al conectar el microcontrolador con una PC vía USB, el equipo de cómputo reconozca al puerto como si se tratase de un puerto serie tipo RS-232 [9].

El archivo ***usb_cdc.h*** incluye todo el respaldo necesario para configurar la conexión USB del microcontrolador y emular dicho puerto como si se tratase de un puerto RS-232. Por lo que para llevar a cabo esta tarea lo primero que se debe hacer es llamar al archivo de biblioteca declarándolo en la cabecera del programa [3]:

```
#include “usb_cdc.h”                          // Se hace el llamado al archivo que sirve
                                              // para la comunicación tipo RS-232 vía USB.
#use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7)  // Configura la comunicación RS-232.
```

Ahora en el programa principal se agrega el siguiente código para dar de alta y realizar el proceso de enumeración de la conexión USB:

```
usb_init();              // Se da de alta y se verifica la conexión física del microcontrolador vía USB.
usb_cdc_init();          // Se inicia la configuración hardware y software que emula la conexión tipo RS-232.
usb_task();              // Se inicia el proceso de configuración y enumeración de la conexión USB.
usb_enumerated();        // Tiene el valor TRUE o FALSE si se ha enumerado o no la conexión USB del
                         // microcontrolador.
```

Para mandar y recibir datos a través de la conexión USB establecida con anterioridad y siguiendo el protocolo establecido para el puerto RS-232 de una PC, se utilizan las siguientes instrucciones:

```
usb_cdc_getc();          // Esta instrucción recibe el carácter o cadena de bits del bus
                         // de recepción de datos, si no hay  datos en el bus,
                         // esperará indefinidamente hasta encontrar alguno.
usb_cdc_kbhit();         // Regresa el valor TRUE o FALSE dependiendo de si existe o no,
                         // algún dato en el bus de recepción.
usb_cdc_putc(k);         // Manda el dato guardado en la variable “k” a través del bus
                         // de transmisión de datos, esperará indefinidamente hasta que el dato
                         // sea recibido satisfactoriamente por el destinatario.
usb_cdc_putc_fast(k);    // Igual que la instrucción anterior, pero no esperará hasta que el dato sea recibido,
                         // sino que éste se perderá si no es tomado por el destinatario.
usb_cdc_putready();      // Regresa el valor TRUE o FALSE si existe o no, espacio en el bus de transmisión de
                         // datos.
```

3.3.2.4. OTRAS ARCHIVOS DE BIBLIOTECA Y DIRECTIVAS IMPORTANTES.

Existen otras llamadas importantes que tienen que ser anexadas en el código para toda aplicación, ya que fungen como parte de la configuración del microcontrolador específico a utilizar y otros detalles del mismo:

```
#include <18F4550.h>          // Llama al archivo encargado del manejo del PIC18F4550.
#fuses HSPLL, NOWDT, NOPROTECT, LVP, NODEBUG, USBDIV, PLL5, CPUDIV1, VREGEN
                              // Configura las interrupciones propias del microcontrolador.
#use delay(clock=20000000)    // Indica que la señal de reloj con la que trabajará el PIC es de 20 MHz.
```

3.4. RESUMEN Y SÍNTESIS.

En este capítulo se detallaron de forma general los lenguajes de programación que serán empleados en la realización del proyecto de tesis. Se describió al ambiente *Microsoft Visual Basic 6.0* como un lenguaje basado en objetos, y orientado a eventos ligados a elementos. También se detalló que este lenguaje permite crear interfaces visuales para facilitar la interacción del usuario con el sistema, y se describieron los controles, comandos y eventos que serán de interés para el proyecto. Esta descripción no excedió la profundidad necesaria para realizar dicha labor y la decisión de no ir más allá en el estudio de dicho lenguaje se debe a que no es el objetivo principal de este texto que el diseñador sea un experto en el manejo de dicha herramienta.

De la misma forma se detallaron las características del lenguaje *CCS PIC C Compiler*. Esta herramienta de programación es un tipo de "*cross-compiler*" ya que los programas se editan y prueban con ayuda de una PC para que, después de la compilación, el programa resultante en código máquina sea instalado y ejecutado en un microcontrolador *PIC* de *Microchip, Inc*. De nueva cuenta sin profundizar demasiado se describieron los comandos y archivos de biblioteca y configuración que permiten que el microcontrolador ejecute las funciones vitales para el proyecto, tales como lectura y escritura de puertos, lectura del convertidor A/D y configuración de la comunicación USB emulando el protocolo RS-232.

CAPÍTULO 4.
ESTRUCTURA DEL SISTEMA DE SEGURIDAD.

En este capítulo se describen los elementos que componen un sistema de seguridad, ya que el primer paso para el diseño del mismo es la visión y el completo entendimiento de la arquitectura de un sistema como éste.

La mayoría de los sistemas de seguridad cuentan con diversos sensores colocados tanto en la periferia de la propiedad como en el interior de la misma. Cuentan también con un equipo electrónico de control para la recopilación y procesamiento de las señales de los mismos, y la activación o desactivación de las alarmas, cerrojos, etcétera. Finalmente la última parte de un sistema de seguridad está compuesta precisamente por estos elementos establecidos como periféricos de salida, tales como alarmas sonoras, luces, cerrojos eléctricos o electrónicos, dispositivos de comunicación telefónica, entre otros.

Un sistema de seguridad puede tener una mayor o menor calidad respecto a otro dependiendo de la eficiencia y funcionalidad de los elementos que la conforman, así como la calidad y complejidad de los mismos. Una forma sencilla de entender esto es por ejemplo comparar el tipo de sensores que conforman el sistema de seguridad, existen sensores de muchos tipos, que miden o detectan diferentes propiedades elementales y que inclusive se pueden conectar de forma inalámbrica.

4.1. INTRODUCCIÓN A LOS SISTEMAS DE SEGURIDAD.

La figura 4.1 muestra la estructura de un sistema de seguridad, mediante un diagrama a bloques.

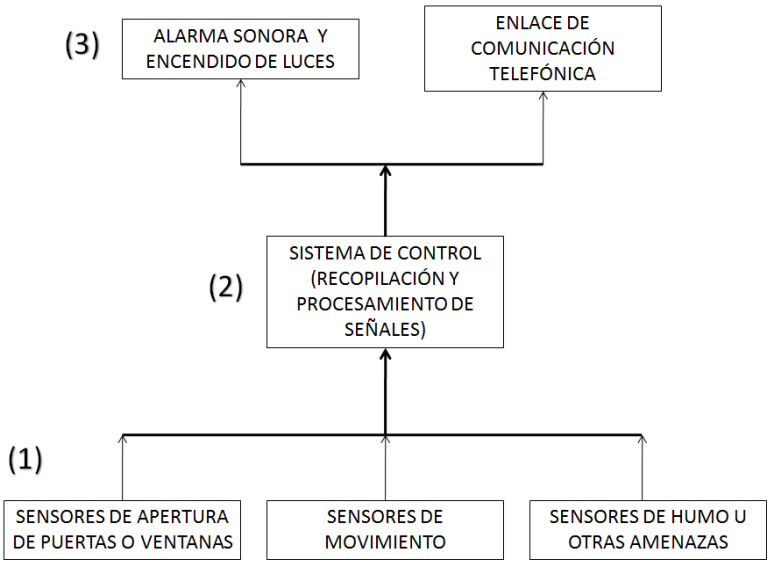


Figura 4.1. Diagrama a bloques de un sistema de seguridad típico.

La primera etapa de un sistema de seguridad es la de ***detección***, es en esta parte donde se obtienen las señales de interés concernientes a posibles amenazas como, intrusión de personas no autorizadas, fuego o fugas de gas, entre muchas otras que pueden ser importantes para el usuario. Para poder detectar esta clase de eventos es necesario hacer uso de diversos tipos de sensores.

La segunda etapa es la de ***control***, aquí se recopilan las señales, se adecuan o procesan las mismas y con base en ellas se toman decisiones que terminan con la ejecución de ciertas acciones que fungen como respuesta del sistema de seguridad a las amenazas que se estén presentando en ese momento, ya que no se ejecutarán las mismas acciones cuando se detecte una amenaza de robo que cuando se detecte una amenaza de incendio.

La tercera y última etapa del sistema es la de ***ejecución*** y está compuesta por los periféricos de salida. Estos pueden ir desde dispositivos simples como luces parpadeantes, hasta dispositivos más complejos como equipos de comunicación telefónica. Éste último es vital para los sistemas comerciales actuales, ya que permiten la comunicación con la propia central de seguridad particular de la empresa o con la central de seguridad pública de la entidad donde se encuentre la propiedad.

El sistema de seguridad desarrollado como proyecto en este texto sigue prácticamente en su totalidad la estructura del diagrama de la figura 4.1.

4.1.1. DESCRIPCIÓN DEL SISTEMA.

Al igual que cualquier sistema de seguridad típico o comercial, éste sistema cuenta con una etapa de detección, otra de control y una más de ejecución.

Como característica particular, el sistema de seguridad desarrollado en este proyecto cuenta con un servicio adicional al convencional, ya que aunque no es considerada una amenaza, el sistema detecta si alguna persona habilita el timbre de la propiedad y es capaz de comunicar vía telefónica a dicha persona con el propietario de la vivienda, con ayuda de un intercomunicador instalado en la misma. Con un propósito meramente práctico el sistema podrá detectar la apertura no autorizada de una puerta o una ventana, para ejemplificar los pasos a seguir en el diseño e instalación de sensores.

La *etapa de control* estará conformada por una PC que trabajará con una aplicación desarrollada en *Visual Basic 6.0* y un microcontrolador *PIC18F4550* que será programado con un código compilado con *PIC C Compiler*. Mientras que el microcontrolador recopila las señales detectadas por los sensores, será la PC la encargada de procesar dichas señales y tomar las decisiones correspondientes. Sin embargo, serán ambos elementos los que se encarguen de activar y desactivar los periféricos de salida, aunque sea la PC la que controle dichas decisiones en su totalidad.

La *etapa de ejecución* como se acaba de describir será comandada por la PC y manejada tanto por la PC como por el microcontrolador. Está parte estará conformada por un teléfono móvil que se encargará de realizar la comunicación con el usuario tanto para informar de activaciones del timbre de la propiedad como para avisar de alguna alerta generada por alguna intromisión que signifique algún intento de robo a la misma. Otro periférico será una chapa electrónica que podrá activarse o desactivarse por medio de una llamada telefónica al celular local del sistema, haciendo uso de los tonos del teclado telefónico.

Si se reflexiona un poco sobre este último periférico de salida, resulta sencillo darse cuenta que el teléfono celular local también tiene un papel importante como periférico de entrada, ya que además de que el sistema informe al usuario sobre los eventos ocurridos en su propiedad, debe ser capaz también de contestar a una llamada hecha por el propietario para la activación o desactivación de la chapa. La figura 4.2 muestra un diagrama a bloques más específico del sistema de seguridad desarrollado.

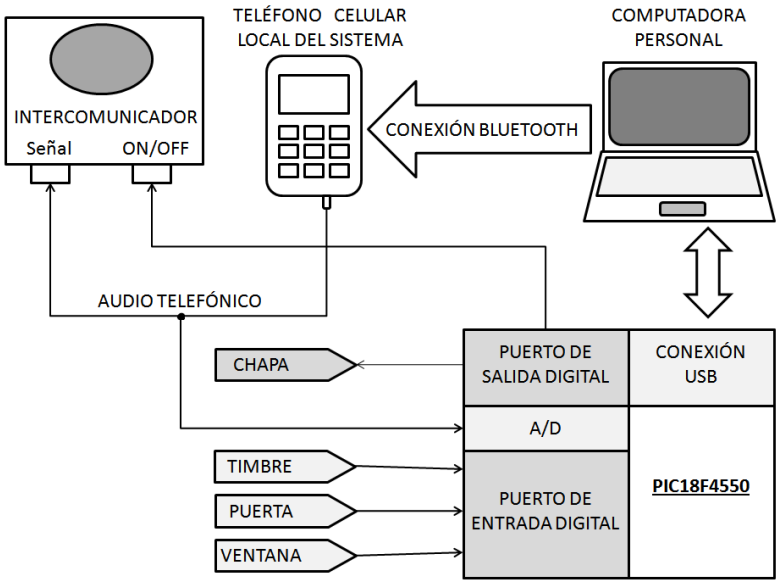


Figura 4.2. Estructura de conexiones del sistema de seguridad.

4.2. SENSORES Y DETECCIÓN.

Para comenzar a describir bloque por bloque la estructura del sistema de seguridad, se debe comenzar con la primera etapa del mismo, es decir la parte dirigida a los sensores que serán empleados, para posteriormente continuar con el proceso de detección con ayuda del microcontrolador. Existen diversos sensores que pueden ser útiles para cumplir con el objetivo de esta etapa, y para elegir el más adecuado es importante tomar en cuenta la disposición geométrica y mecánica de los elementos que se desean monitorear e inclusive el ambiente en el cual se encuentran los mismos, en un ambiente de mucho polvo por ejemplo algún tipo de sensor óptico puede funcionar erróneamente o inclusive dejar de funcionar.

4.2.1. SENSORES DEL SISTEMA.

Para efectos prácticos se ha decidido utilizar como sensores un par de interruptores opto-electrónicos infrarrojos de manera que con ayuda de éstos se pueda detectar de forma simple la apertura de una puerta o una ventana. Este tipo de sensores consta de dos elementos básicos, un led infrarrojo y un fototransistor que detecta la presencia de radiación en esa longitud de onda. Ambos elementos se encuentran instalados en una coraza plástica común, de manera que se encuentren alineados exactamente uno frente al otro, dicha coraza posee una apertura de manera que algún elemento pueda obstruir por algún efecto mecánico el paso de radiación hacia el fototransistor proveniente del led infrarrojo. De esta manera el abrir o cerrar

una puerta o cualquier otro artefacto puede provocar como consecuencia el efecto antes descrito y por ende obtener como salida dos valores lógicos diferentes en el estado de corte o saturación del fototransistor. La figura 4.3 muestra una imagen del opto-interruptor *H21A1* y su diagrama esquemático.

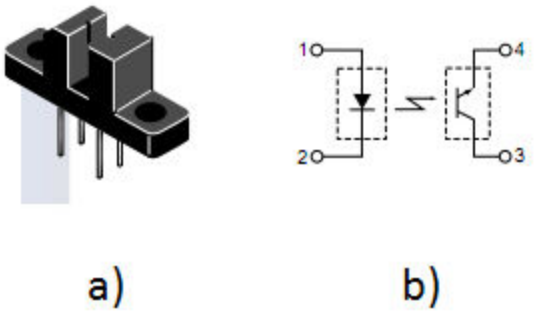


Figura 4.3. a) Opto-interruptor H21A1. b) Diagrama esquemático.

La figura 4.4 muestra un diagrama de la forma en que deben ser polarizados los elementos y conectadas las terminales de cada uno para que el microcontrolador pueda detectar la señal de salida del sensor, ésta debe variar entre dos niveles lógicos de voltaje *+5 V* y *GND*. Una forma de lograr dicho objetivo es polarizando el circuito con *Vcc=5 V*, *R1* podría tener un valor aproximado *330 Ω* y *R2* uno de aproximadamente *10 KΩ*, de esta manera al obstruir o permitir el paso de la radiación con algún objeto de material denso a través de la franja de obstrucción, se obtendrán a la salida los dos niveles de voltaje antes mencionados dependiendo de la situación.

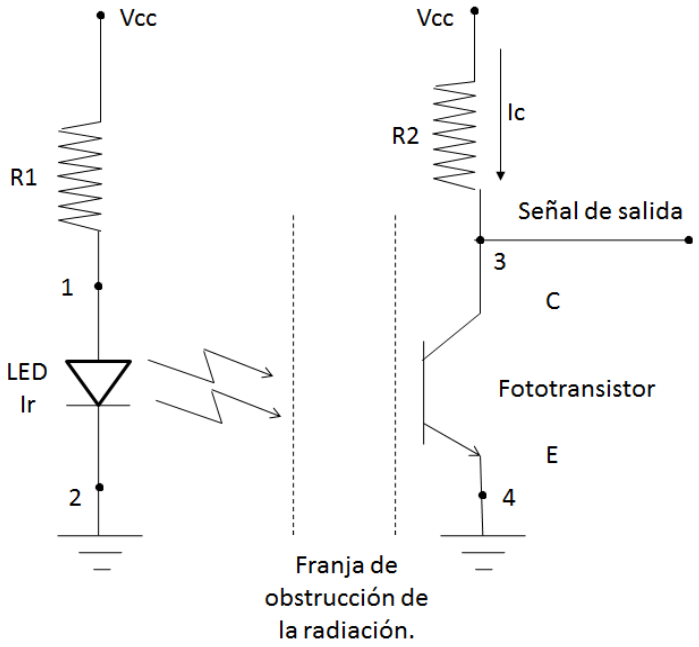


Figura 4.4. Diagrama de conexión del sensor.

4.2.2. PROCESO DE DETECCIÓN.

Bajo las condiciones mencionadas en el punto anterior, describir el proceso de detección en el cambio de estado de la señal de salida resulta relativamente simple. Mientras el opto-interruptor se encuentra correctamente polarizado con un voltaje *Vcc=5 V* a través del led se encuentra circulando una corriente que produce que éste dispositivo emita radiación infrarroja. De la hoja de especificaciones del opto-interruptor se puede observar que el voltaje de encendido del diodo emisor es de alrededor de *1.5 V* por lo que en la resistencia *R1* se estará presentando un voltaje de aproximadamente *3.5 V* y la corriente que estará circulando a través de ambos elementos será de *10.61 mA*, tal como se muestra en la ecuación (4.1). Esta corriente se encuentra muy por debajo de la máxima establecida para el emisor según la hoja de datos del *H21A1*, lo que implica que el emisor del opto-acoplador está trabajando en una zona de bajo consumo y riesgo.

$$I_E = \frac{V_{R_1}}{R_1} = \frac{V_{cc} - V_D}{R_1} = \frac{5\ V - 1.5\ V}{330\ \Omega} = \frac{3.5\ V}{330\ \Omega} = 10.61\ mA \tag{4.1}$$

De esta manera cuando no hay ningún objeto en la rendija que obstruya el paso de radiación hacia el fototransistor, éste se encontrará en estado de saturación con una corriente circulando a través del colector de aproximadamente *0.5 mA* ya que el voltaje entre colector y emisor tenderá a ser prácticamente cero (V_{CEsat}*=0 V*). Cuando algún objeto se encuentre interrumpiendo el paso de radiación hacia el detector, la corriente a través del colector y la resistencia *R2* será prácticamente cero y el voltaje entre el colector y el emisor será de *5 V*. Las ecuaciones (4.2) y (4.3) describen dichos efectos.

$$I_{C_{sat}} = \frac{V_{cc} - V_{CE_{sat}}}{R_2} = \frac{5\ V - 0\ V}{10\ K\Omega} = 0.5\ mA \tag{4.2}$$

$$V_{CEcte} = V_{cc} - R_2 I_{co} = 5\ V - (10\ K\Omega)(0) = 5\ V \tag{4.3}$$

La tabla 4.1 muestra una comparación entre los valores reales medidos en el laboratorio y los valores teóricos desarrollados anteriormente:

	Valor teórico	Valor medido
V_{CEsat}	0 V	0 V
I_{Csat}	0.5 mA	0.469 mA
V_{CEcte}	5 V	4.985 V
I_{Ccte}	0 mA	0.008 mA

Tabla 4.1. Resultado de la medición de voltajes y corrientes en el opto-interruptor.

Existen muchas marcas y modelos de opto-interruptores, y cada uno puede variar en la forma en que responde al manejo aquí descrito (corriente y voltaje), así como en sus máximos valores de funcionamiento.

4.3. SISTEMA DE CONTROL.

Todo sistema de seguridad posee una etapa de control que se encarga de recopilar y procesar las señales provenientes de los sensores y enviar las señales respectivas de habilitación para los periféricos de salida. A continuación se describe qué elementos conforman dicho sistema y como podrán configurarse y conectarse dichos elementos para alcanzar su objetivo fundamental, además se tratarán algunos diagramas de flujo y algoritmos que permitan conceder al lector una idea clara del proceder de los elementos del sistema de control.

4.3.1. PROPIEDADES DEL MICROCONTROLADOR PIC18F4550.

El elemento encargado de recopilar las señales de los sensores y capturar el audio del teléfono celular es un microcontrolador *PIC18F4550*, éste es fabricado por la compañía *Microchip, Inc*. Posee una gran variedad de unidades funcionales embebidas como temporizadores, unidades de comparación, captura y PWM, convertidores analógico-digital y posiblemente por lo que es más reconocido es por su capacidad de comunicación avanzada USB 2.0.

La figura 4.5 muestra el diagrama de pines del microcontrolador PIC18F4550, con sus respectivas funcionalidades [9].

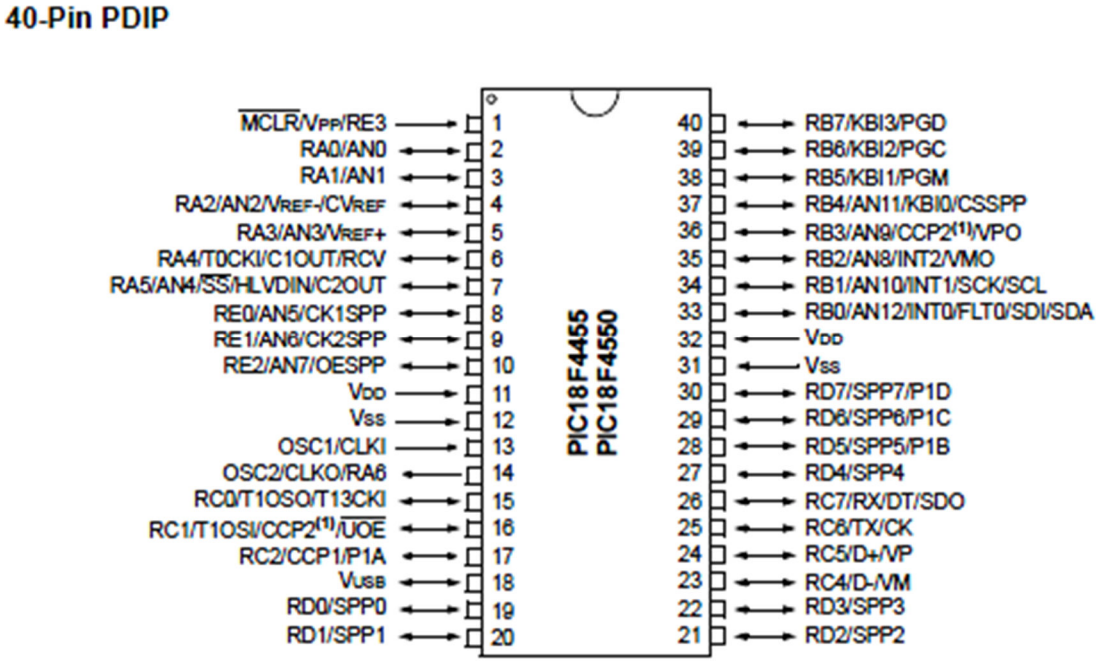


Figura 4.5. Diagrama de pines del *PIC18F4550* con encapsulado PDIP.

Tiene una arquitectura tipo *RISC* avanzada *Harvard* con un conjunto reducido de 75 instrucciones en lenguaje ensamblador, otras características de dicho microcontrolador son [9]:

- 40 pines (para el modelo de encapsulado del chip a utilizar).
- 32 KBytes de memoria de programa.
- 2 KBytes de RAM y 256 Bytes de EEPROM.
- Multiplicador hardware 8x8.
- Frecuencia de reloj máxima de 48 MHz.

Para poder programar y hacer uso de dicho microcontrolador se empleará la tarjeta de desarrollo para proyectos didácticos UPRsys V0308 mostrada en la figura 4.6 [10]. Esta tarjeta fue diseñada y manufacturada por el *departamento de ingeniería mecatrónica* de la *facultad de ingeniería* de la *UNAM* y permite trabajar y cargar programas a este tipo de microcontroladores con ayuda del software *Bootloader PICDEM FS USB v1.0* desarrollado por *Microchip, Inc*.

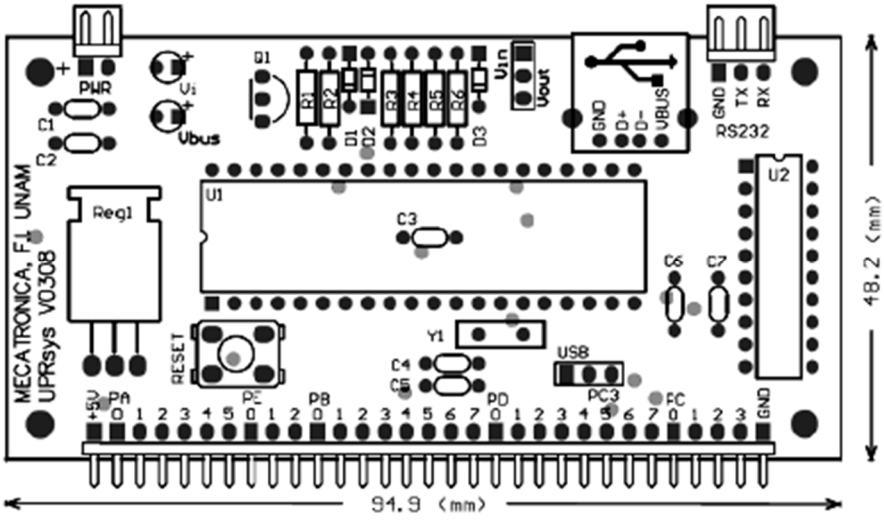


Figura 4.6. Vista de la tarjeta UPRsys V0308.

4.3.2. RECOPILACIÓN DE LAS SEÑALES DE LOS SENSORES.

Como ya se mencionó, el microcontrolador será el encargado de recopilar las señales de los sensores y transmitir dichos datos a la PC vía el puerto USB. Puesto que las señales provenientes de los sensores serán de tipo digital, o más en concreto de tipo binario puesto que solo podrán tomar dos valores lógicos posibles, es necesario captar o leer dichas señales a través de un puerto digital.

Como la tarjeta de desarrollo provee al puerto D del microcontrolador de completa libertad funcional en todos su pines, se elegirá a éste como puerto de entrada digital para realizar la labor de lectura y recopilación de las señales de los sensores. La razón de que se utilice un solo puerto para este trabajo se debe básicamente a la optimización del algoritmo de detección de los sensores que se describirá a continuación.

La figura 4.7 muestra un diagrama a bloques del algoritmo de detección de los sensores únicamente, más adelante se detallará el algoritmo correspondiente a la detección del audio del teléfono celular y los tonos DTMF. Como puede observarse solo es necesario leer el puerto una sola vez por ciclo o iteración, en cambio si tuviéramos que leer de pin en pin, el proceso de lectura y envío de dato se realizaría hasta ocho veces antes de comenzar de nueva cuenta el ciclo.

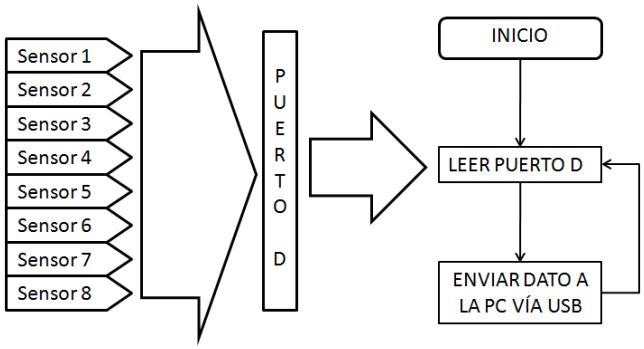


Figura 4.7. Algoritmo de detección de los sensores.

La figura 4.7 muestra además que una vez que el sistema arranca, el proceso de lectura del puerto se realiza a través de un ciclo infinito, lo que quiere decir que el sistema estará verificando el estado de los sensores de forma continua y enviando un dato correspondiente a la lectura hacia la PC para la posterior toma de decisiones del sistema. El código editable en lenguaje C para PIC sería de la siguiente forma:

```
char sensor;                  // Se declara la variable "sensor" que puede tomar valores de 0 a 255.
while(true){                  // Se comienza un ciclo infinito.
sensor=input_D();             // Se lee el valor del puerto D correspondiente a un estado de los sensores.
usb_cdc_putc_fast(sensor);    // Se envía por el puerto USB el valor del puerto D del microcontrolador.
}
```

Por supuesto en el código mostrado anteriormente no se agregan las instrucciones que indican el inicio del programa principal *main()*, ni los archivos de biblioteca e instrucciones que permiten la activación y configuración del puerto USB y la comunicación *usb-cdc* ya que en el capítulo 3 se detallaron más a fondo dichas herramientas.

Otro detalle que es importante mencionar acerca de este algoritmo es el hecho de que el puerto D puede tomar hasta 256 valores diferentes, dicho factor deberá ser tomado en cuenta más adelante en la toma de decisiones del sistema.

4.3.3. ENVÍO Y RECEPCIÓN DE DATOS ENTRE EL MICROCONTROLADOR Y LA PC.

Hasta ahora se ha descrito la comunicación vía USB entre el microcontrolador y la PC de manera muy poco profunda. Dicho proceso aunque no es nada complejo, puede ser imposible de llevar a cabo si los dos dispositivos que se desean comunicar no se encuentran propiamente configurados, por lo que esta labor se describirá en el próximo capítulo.

En el capítulo 3 de este texto se describen las instrucciones y archivos que permiten la comunicación vía USB emulando un puerto serie del tipo RS-232, tanto en PIC C de CCS para el microcontrolador como en Visual Basic 6.0 de Microsoft para la PC, por lo que toca ahora especificar más a fondo el código necesario para que el algoritmo descrito en el subtema anterior pueda llevarse a cabo.

Si se observa de nuevo el diagrama de flujo mostrado en la figura 4.7, se puede notar que el proceso de lectura y envío de datos hacia la PC se realiza dentro de un ciclo infinito, por lo tanto la PC debe estar recibiendo dichos datos continuamente en cuanto el sistema arranque. Como el programa diseñado para el equipo de cómputo es desarrollado en Visual Basic 6.0, y tal como se explicó en el capítulo 3, este lenguaje es orientado a eventos, por lo que se deberá hacer uso del control *Timer* de dicho lenguaje para poder monitorear de manera continua y periódica la recepción de datos enviados desde el microcontrolador.

A continuación se muestra el código necesario para la configuración del puerto USB del microcontrolador y el proceso completo de lectura de puerto y envío de datos hacia la PC:

```
#include <18F4550.h>          // Se llaman a los archivos de biblioteca del PIC19F4550.
#fuses HSPLL,NOWDT,NOPROTECT,LVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=20000000)  // Se indica que se trabajará con una señal de reloj de 20 MHz.
#use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7)  // Se configura la comunicación RS-232.
#include "usb_cdc.h"          // Se habilita la comunicación USB emulando al puerto RS-232.
void main(void)               // Inicio del programa principal.
{
usb_init();                   // Se da de alta y se verifica la conexión física del microcontrolador vía USB.
usb_cdc_init();               // Se inicia la configuración hardware y software
                              // que emula la conexión tipo RS-232.
usb_task();                   // Se inicia el proceso de configuración y enumeración de la conexión USB.
do                            // Se inicia un ciclo de encendido y apagado para un led conectado
{                             // al pin 0 del puerto C con el objetivo de que el usuario pueda enterarse
                              // de forma visual del estado del puerto.
output_high(PIN_C0);          // Se pone en valor alto el pin C0.
```

```
  delay_ms(300);            // Se indica un retardo de 300 mseg. antes de ejecutar la siguiente instrucción.
  output_low(PIN_C0);       // Se pone en valor bajo el pin C0.
  delay_ms(300);            // Se indica un retardo de 300 mseg. antes de ejecutar la siguiente instrucción.
}                           // El ciclo termina cuando el puerto es enumerado correctamente.
while(!(usb_enumerated()==TRUE));
output_high(PIN_C0);        // Como resultado el pin C0 se mantiene en adelante con un valor lógico alto.
char sensor;                // Se declara la variable "sensor" que puede tomar valores de 0 a 255.
while(true){                // Se comienza un ciclo infinito.
 sensor=input_D();          // Se lee el valor del puerto D correspondiente a un estado de los sensores.
 usb_cdc_putc_fast(sensor); // Se envía por el puerto USB el valor del puerto D del microcontrolador
}                           // El ciclo comienza de nuevo sin esperar que el dato haya sido adquirido por la PC.
```

La instrucción *usb_cdc_putc_fast()* se utiliza en lugar de la instrucción *usb_cdc_putc()* ya que ésta última aguarda hasta que el dato sea adquirido por la PC. La justificación radica básicamente en que el ciclo en el que se ejecuta todo este proceso sucede de forma relativamente rápida y la PC puede adquirir los datos prácticamente en el momento en que lo solicita, además puesto que el microcontrolador será capaz de detectar también los tonos DTMF del audio telefónico es necesario que el ciclo de detección se vea lo menos retardado posible por cuestiones de comunicación.

En Visual Basic 6.0 se realizó el programa para que la PC pueda recibir los datos enviados desde el microcontrolador. El primer paso es crear un nuevo proyecto, para agregar todos los controles que nos permitan generar una interfaz visual para cualquier usuario, y entre ellos se encuentra por supuesto el control *Timer* del que ya se han detallado sus características básicas con anterioridad. Como cualquier nuevo proyecto, se partirá de un formulario estándar en el cual se agregarán únicamente tres controles para este ejemplo: una *caja de texto*, un control *MSComm* y un control *Timer*. Para este ejemplo no cambiaremos los nombres que vienen por default en los elementos que conforman el proyecto, es decir, el nombre del formulario estándar será *form1*, el de la caja de texto *Text1*, el del control de comunicación serie MSComm1 y el del control *Timer* será *Timer1*.

El primer paso es configurar la comunicación con el puerto USB emulando una conexión de tipo RS-232, los *drivers* del microcontrolador se encargan de dicha labor, de forma que la PC detecta dicho dispositivo y le asigna un puerto COM de comunicación serie para establecer la conexión con el mismo, por lo tanto en el código de Visual Basic solo es necesario dar de alta dicha conexión. Para llevarlo a cabo, se puede ocupar como evento que desencadene la habilitación en la comunicación, la propia carga del formulario principal, para ello solo es necesario dar un doble clic en el formulario para acceder a la ventana de edición de código y anexar en la misma los siguientes comandos:

```
Private Sub Form_Load()  ' El evento que permite que los siguientes comandos se ejecuten es la carga del
                         ' formulario.
MSComm1.CommPort=1       'Se indica que el puerto COM con el que se establecerá la conexión es el número 1.
```

```
MSComm1.Settings="115200,N,8,1"          'Se configuran las características del protocolo
                                         'de comunicación serie.
MSComm1.InputMode=comInputModeText       'La comunicación se trabajará con caracteres en código ASCII.
MSComm1.PortOpen=True                    'Se habilita y abre el puerto bajo los parámetros configurados.
End Sub
```

Cuando el usuario termina la aplicación, es decir que cierra el programa, todos los puertos habilitados son cerrados automáticamente y por ende se termina con la comunicación entre la PC y los dispositivos involucrados en este caso el microcontrolador. Para el ejemplo que estamos describiendo se modela que el puerto COM asignado a la conexión USB del microcontrolador sea el número 1. Ahora teóricamente ya se podrían enviar y recibir datos a través de dicho puerto de comunicación.

Puesto que el objetivo es que el usuario no tenga que estar interviniendo en el proceso de comunicación entre el microcontrolador y la PC, además de que este proceso debe estarse llevando a cabo de manera periódica y continua, el control *Timer* será el encargado de realizar la labor de recepción de datos recopilados por el microcontrolador. Para ello solo es necesario dar doble clic en el control *Timer1* y de esta forma acceder a su ventana de edición de código para anexar los siguientes comandos:

```
Private Sub Timer1_Timer()
sensor = MSCommVirtual.Input    'Se lee y guarda en la variable sensor el dato recibido.
Text1.Text = sensor             'Se despliega en la caja de texto el carácter correspondiente
 End Sub                        'al valor leído en código ASCII.
```

Tal como se explica en el capítulo 3, estos comandos se estarán ejecutando periódicamente con un lapso de tiempo determinado por la propiedad *Interval* del control *Timer*, ésta se puede modificar tanto en tiempo de diseño como en tiempo de ejecución, y para este ejemplo representará el periodo de tiempo entre lectura de datos provenientes del microcontrolador. Este intervalo se configura en milisegundos y entre mayor sea dicho lapso, la respuesta del sistema a cambios de estado en los sensores será más lenta, y por consiguiente para valores muy pequeños de esta propiedad, la respuesta se volverá casi inmediata a la percepción del usuario. La figura 4.8a muestra la vista de una posible ventana en tiempo de diseño para el ejemplo anterior. La figura 4.8b muestra la misma ventana pero en tiempo de ejecución.

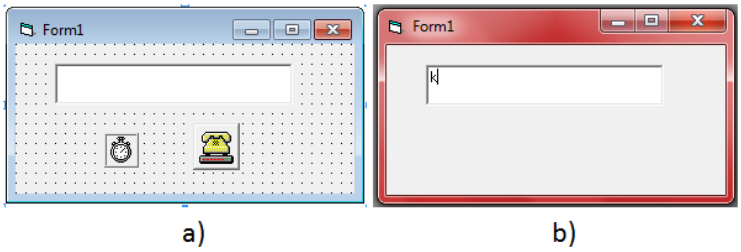


Figura 4.8. Ventana de aplicación para la lectura de datos provenientes del microcontrolador.
a) En tiempo de diseño. b) En tiempo de ejecución.

4.3.4. ALGORITMO DE TOMA DE DECISIONES.

Todo sistema de seguridad responde con base en cambios de estado de los sensores de los que dispone. Puesto que en este sistema todos los sensores se encuentran conectados en un solo puerto, éste responderá a cambios en el valor de estado estable del mismo, es decir, cuando el sistema no capta ninguna amenaza que produzca un cambio de estado en algún sensor, el puerto mantiene por default un valor constante que corresponde al estado inicial y estable del sistema de seguridad (la propiedad no se encuentra amenazada o comprometida de ninguna manera). Ahora bien, si existe un cambio en el estado de algún sensor produciendo así un cambio en el valor del puerto de entrada digital, el sistema debe ser capaz con base en un algoritmo pre-programado de tomar decisiones que permitan una respuesta del mismo sistema a las amenazas o eventos que se estén presentando.

La respuesta del sistema puede ir desde prender y apagar una luz, activar una alarma sonora o hasta realizar una llamada telefónica al usuario o a la estación de policía local, todo dependerá del tipo de evento que se esté llevando a cabo.

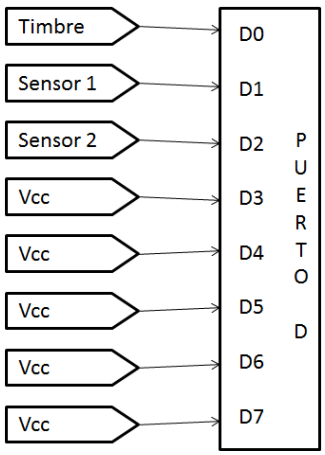


Figura 4.9. Orden de conexión de los sensores y el timbre de la propiedad.

Puesto que inclusive pueden presentarse cambios de estado en más de un sensor, resulta muy importante que el sistema posea un algoritmo de toma de decisiones que permita responder de la manera más efectiva a los diversos eventos o amenazas que pudieran presentarse.

Para lograr esto es muy importante normalizar la forma en que serán conectados los sensores y el resto de periféricos de entrada del sistema. La figura 4.9 muestra el orden en que se conectarán el timbre y los dos sensores ópticos al puerto D del microcontrolador.

De la figura 4.9 puede notarse que sólo tres pines del puerto serán utilizados para este proyecto, el resto serán conectados a tierra para establecer un valor lógico bajo y evitar que el ruido o interferencia puedan distorsionar el valor de la lectura. De esta manera, además del valor inicial y estable del puerto, pueden presentarse otros siete valores diferentes, que tendrán que ser procesados por el sub-sistema de control para tomar una decisión cuando se presente uno de ellos.

La tabla 4.2 muestra todos los valores posibles del puerto con los respectivos eventos que producen dicho valor, además se anexa la respuesta primordial del sistema a dicho acontecimiento. Como puede observarse se especifican dos diferentes combinaciones por número de evento, uno para manejo de la lectura del puerto con lógica positiva y otro para lógica negativa dependiendo de cómo se quiera detectar el cambio de estado de los sensores de su estado estable a su estado de activación.

Puede notarse también que a partir del posible evento número dos el sistema responderá de la misma manera. Esto se debe a que el número de sensores conectados es muy reducido para los fines didácticos de este proyecto, pero de haber conectado un número mayor de sensores, el sistema tendría un número mayor de eventos y poseería una variedad de respuestas más grande. Por ejemplo, si al sistema se le conectara un sensor que detectara la presencia de humo, de manera que se buscara prevenir o combatir un posible incendio en la propiedad, se tendría que agregar al sistema una respuesta en la que se buscaría contactar primordialmente al cuerpo de bomberos local para informar de dicha amenaza.

Para efecto de pruebas el sistema desarrollado llamará al usuario en lugar de la estación de policía local. Además un poco más adelante, cuando se describa el proceso de interacción remota entre el usuario y el sistema de seguridad, se detallarán las ventajas que posee esta importante propiedad del sistema desarrollado en este proyecto.

Por ahora se puede esbozar un diagrama de flujo del algoritmo de control del sistema de seguridad, dicho diagrama se muestra en la figura 4.10. Nótese que aún no se ha agregado el proceso de reconocimiento de los tonos DTMF del sistema a la hora de establecer una llamada telefónica, la razón de esto es que este módulo del sistema puede verse tanto como periférico

de entrada o como periférico de salida, este es el elemento clave de la interacción remota entre el usuario y el sistema de seguridad y la primordial innovación de este proyecto.

Valor en lógica positiva D_2 D_1 D_0	Valor en lógica negativa D_2 D_1 D_0	Número de evento ocurrido
0 0 0	1 1 1	0. No ha ocurrido ningún evento y la propiedad se encuentra libre de cualquier amenaza detectable.
0 0 1	1 1 0	1. El timbre de la propiedad se ha activado. Se procede a efectuar una llamada telefónica al usuario para la posible activación del intercomunicador.
0 1 0	1 0 1	2. El sensor de la puerta se ha activado, existe una posible intromisión a la propiedad. Se procede a encender la sirena y luces de alarma, y según la configuración del usuario a realizar una llamada telefónica al mismo o a la estación de policía local.
0 1 1	1 0 0	3. Se ha activado tanto el timbre de la propiedad como el sensor de la puerta. Se procede de la misma forma que en el evento número dos por ser prioridad la posibilidad de robo.
1 0 0	0 1 1	4. El sensor de la ventana se ha activado. De igual forma se procede como en el evento número dos.
1 0 1	0 1 0	5. El timbre y el sensor de la ventana se han activado. Por prioridad se procede como en el evento número dos.
1 1 0	0 0 1	6. Los sensores de la puerta y ventana se han activado. Se procede como en el evento número dos.
1 1 1	0 0 0	7. Tanto el timbre como los sensores están activados. Se procede como en el evento número dos.

Tabla 4.2. Descripción de posibles eventos caracterizados por los diferentes valores del puerto de entrada.

Cabe mencionar que para este proyecto se utilizó el procesamiento de los sensores con lógica negativa.

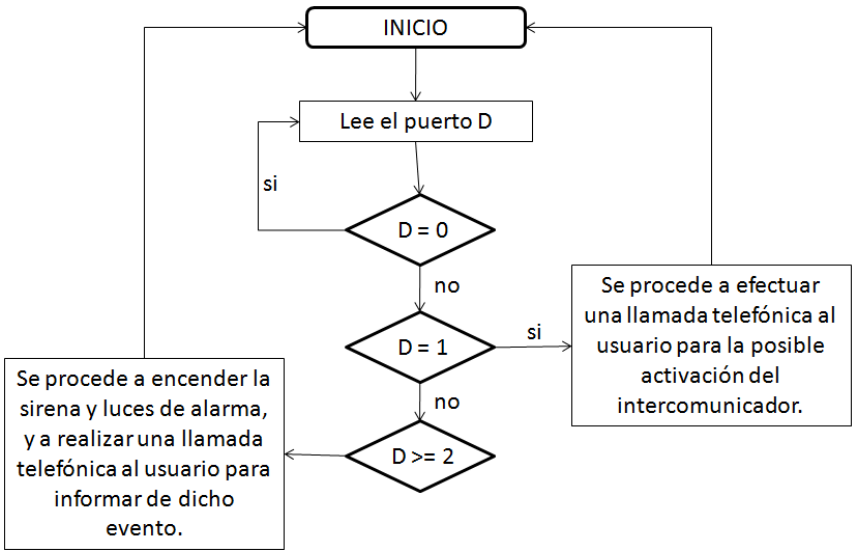


Figura 4.10. Diagrama de flujo de la toma de decisiones.

4.4. PROCESAMIENTO DIGITAL DE AUDIO Y DETECCIÓN DE TONOS DTMF.

El módulo encargado del procesamiento del audio del teléfono celular local es la parte esencial del sistema, con base en él es posible que el usuario interactúe de forma remota con el sistema de seguridad de la propiedad. Con tan solo presionar las teclas de su teléfono particular o llamando al teléfono celular local del sistema, éste será capaz de responder a instrucciones y respuestas codificadas por medio de tonos DTMF que viajan a través del canal de comunicación telefónica cuando una llamada se está llevando a cabo.

Ahora la telefonía móvil procesa las llamadas telefónicas de forma digital, por lo que los teléfonos celulares transforman la señal analógica adquirida por la voz de uno de los interlocutores a una señal digital para que de esta manera sea transmitida por la red de comunicación, no sin antes adecuar, codificar y modular dicha señal para que la transmisión sea lo más eficiente posible. El teléfono del otro interlocutor recibe entonces esta señal digital y procede a demodular, decodificarla y convertirla de nueva cuenta en una señal analógica que es proporcional a la adquirida por el teléfono del primer interlocutor. Este proceso por supuesto es bidireccional y es llevado a cabo en tiempo real, por lo que los equipos celulares que intervienen en el mismo contienen dispositivos de procesamiento digital de señales DSPs (*Digital Signal Processors*) que tienen capacidad de procesamiento a grandes velocidades.

Como el sistema de seguridad se encuentra conformado por dispositivos o elementos que trabajan de forma digital, como lo son la PC y el microcontrolador, es necesario convertir de nueva cuenta las señales de audio del teléfono local en señales digitales que puedan ser

procesadas por los elementos antes mencionados. El objetivo del procesamiento digital de estas señales es detectar la presencia de tonos DTMF, y para ello será necesario el empleo de técnicas fundamentales en la teoría de procesamiento digital como son por ejemplo el teorema de muestreo y la transformada discreta de Fourier (DFT).

4.4.1. TEOREMA DEL MUESTREO.

Como ya se mencionó la conversión análogo-digital es vital para el acoplamiento del audio del teléfono local con el sistema. Esto se logra recopilando muestras de dicha señal analógica a intervalos de tiempo específicos, a este proceso se le llama *muestreo*. Las muestras recopiladas conforman entonces una señal discreta, dichas muestras contienen diferentes magnitudes y son cuantizadas a valores digitales basados en la longitud de una palabra de L bits, a este sub-proceso de muestreo se le llama *cuantización* [2].

El análisis matemático del proceso de muestreo puede llevarse a cabo de la siguiente manera. Siendo *x(t)* una señal continua limitada en banda si se toman muestras a intervalos definidos con un periodo *T* por medio de la multiplicación de la señal *x(t)* con un tren de impulsos unitarios en el tiempo continuo $\delta_T(t)$, como resultado se obtendrá una señal discreta $x_s(t)$ correspondiente a la señal *x(t)* muestreada [2].

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{4.4}$$

$$x_s(t) = x(t) \cdot \delta_T(t) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \delta(t - nT) \tag{4.5}$$

Es importante recalcar que la señal *x(t)* en el dominio de la frecuencia también se ve alterada debido al proceso de muestreo, puesto que la multiplicación en el dominio del tiempo de dos señales equivale a la convolución en el dominio de la frecuencia de dichas señales [2].

$$X_s(\omega) = X(\omega) * \delta_T(\omega) \tag{4.6}$$

El *teorema de muestreo* o *teorema de Nyquist* establece que para evitar la pérdida de información y poder reconstruir la señal original a partir de la señal muestreada, la razón de muestreo debe ser al menos dos veces mayor que el ancho de banda la señal original [2].

$$f_s \geq 2f_x \tag{4.7}$$

Donde *fs* es la frecuencia de muestreo y f_x es la frecuencia máxima de la señal *x(t)*.

La convolución de *X(ω)* con un tren de impulsos $\delta(\omega)$ corresponde a la suma de repeticiones de *X(ω)* centradas en cada impulso desplazado $n\omega_s$, utilizando la transformada de Fourier y el teorema de la convolución para $x_s(t)$ [2].

$$X_s(\omega) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X(\omega - n\omega_s);\ \ \omega_s = \frac{2\pi}{T} \tag{4.8}$$

Donde ω_s es la frecuencia angular de muestreo y *T* el periodo de muestreo.

Por lo tanto cuando se aplica el teorema de muestreo se produce que las repeticiones del espectro no se solapen, evitando de esta manera la superposición de frecuencias y por ende la pérdida de información. Dicho efecto se ilustra en la figura 4.11. La figura 4.11a ilustra el dominio en frecuencia de la señal continua *x(t)*. La figura 4.11b muestra el dominio en frecuencia de un tren de impulsos $\delta_T(t)$. La figura 4.11c ilustra el dominio en frecuencia de la señal discreta $x_s(t)$ cuando $f_s=2f_x$, siendo f_x la máxima frecuencia presente en la señal *x(t)*. La figura 4.11d muestra el traslape que se presenta cuando $f_s<2f_x$, las áreas grises muestran dicha región que por lo tanto puede producir pérdidas de información. Por último la figura 4.11e ilustra el dominio en frecuencia de la señal $x_s(t)$ cuando $f_s>2f_x$, en este caso también se está cumpliendo con el teorema de muestreo antes mencionado.

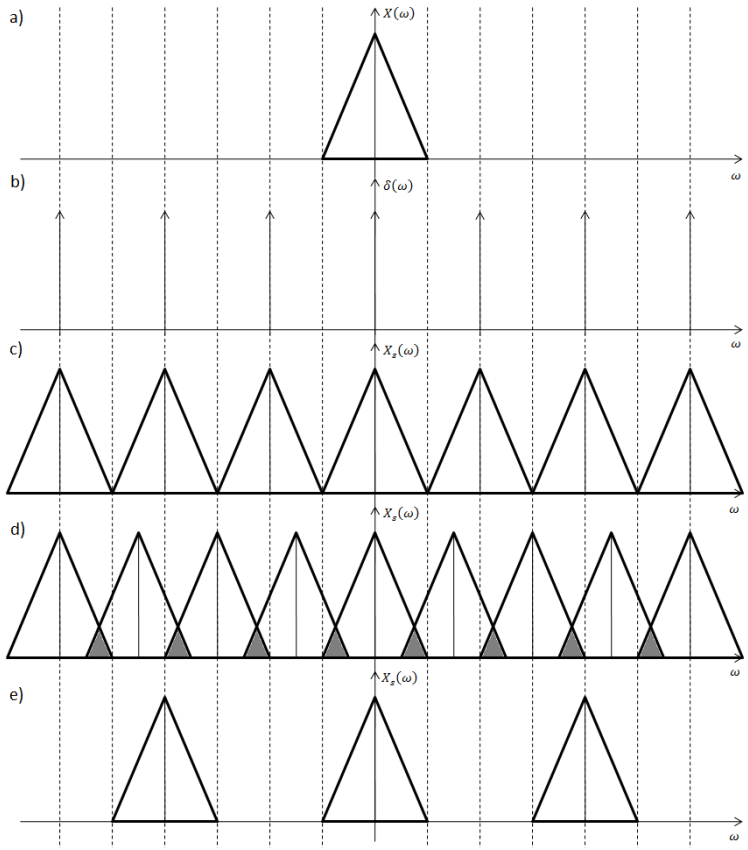


Figura 4.11. a) X(ω) b) $\delta_T(\omega)$ c) $X_s(\omega)$ cuando $\omega_s=2\omega_x$ d) $X_s(\omega)$ cuando $\omega_s=1.5\omega_x$ e) $X_s(\omega)$ cuando $\omega_s=4\omega_x$

4.4.2. LA TRANSFORMADA DISCRETA DE FOURIER.

La transformada discreta de Fourier es una herramienta muy importante en el análisis de señales, ya que nos permiten conocer sus componentes en frecuencia, lo cual funge como objetivo principal para esta aplicación.

La *transformada discreta de Fourier* (DFT), al igual que en su forma continua, es un proceso reversible. También posee las características de periodicidad y simetría. Para una señal *x(n)* acotada, la transformada discreta de Fourier se calcula para una cantidad de *N* muestras, siendo esta cantidad *N* igual al periodo de dicha transformada que además es simétrica con respecto a N/2. La DFT está definida por la siguiente ecuación [2]

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(\frac{2\pi kn}{N})} \tag{4.9}$$

Donde *k* es el índice del espectro discreto desde *k=0* hasta *k=N-1*, y *n* es el índice discreto del tiempo (equivalente al índice de la muestra de la señal) desde *n=0* hasta *n=N-1*.

Puede notarse claramente que el resultado de este proceso estará representado por una cantidad compleja, que puede trabajarse tanto en forma cartesiana como en forma fasorial.

$$X(k) = X_R(k) + jX_I(k) \tag{4.10}$$

$$X(k) = |X(k)| \ \underline{|\emptyset(k)} \tag{4.11}$$

Y que pueden obtenerse con base en las identidades de Euler

$$X_R(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi nk}{N}\right) \tag{4.12}$$

$$X_I(k) = \sum_{n=0}^{N-1} x(n) \, sen\left(\frac{2\pi nk}{N}\right) \tag{4.13}$$

Por lo tanto, la magnitud del espectro es

$$|X(k)| = \sqrt{[X_R(k)]^2 + [X_I(k)]^2} \tag{4.14}$$

Y la fase

$$\emptyset(k) = \tan^{-1}\left({X_I(k)}/{X_R(k)}\right) \tag{4.15}$$

Del análisis anterior, las ecuaciones 4.12, 4.13 y 4.14 serán de vital importancia para el proyecto, puesto que serán implementadas en el algoritmo encargado de ejecutar la transformada discreta de Fourier de las muestras provenientes del audio del teléfono local del sistema de seguridad. Un algoritmo encargado de encontrar los máximos de una serie de datos será usado posteriormente para detectar qué tono dual DTMF se encuentra presente en el audio del teléfono.

4.4.3. PROPIEDADES DE LOS TONOS DTMF.

Una señal DTMF es la suma de dos tonos puros provenientes de dos grupos, un grupo de tonos bajos y otro de tonos altos, cada grupo está compuesto por cuatro tonos individuales distintos. Las frecuencias de los tonos fueron cuidadosamente seleccionadas de tal forma que sus armónicos no se encuentren relacionados y que los productos de su intermodulación produzcan un deterioro mínimo en la señalización. Este esquema contiene dieciséis combinaciones diferentes de las cuales diez representan los números del 0 al 9, y el resto son usados para representar a los caracteres *, #, A, B, C y D. La mayoría de los teclados telefónicos contienen únicamente doce interruptores correspondientes a los números (0-9) más las teclas de asterisco (*) y símbolo numérico (#). Estos se encuentran organizados en una matriz que caracteriza los tonos, uno bajo por renglón y otro alto por columna, de los cuales está compuesto. La figura 4.12 muestra el arreglo matricial que caracteriza a los tonos DTMF.

	1209 [Hz]	1336 [Hz]	1477 [Hz]	1633 [Hz]
697[Hz]	1	2	3	A
770 [Hz]	4	5	6	B
852 [Hz]	7	8	9	C
941 [Hz]	*	0	#	D

Figura 4.12. Matriz característica de los tonos duales de multi-frecuencia (DTMF).

Los tonos referidos como A, B, C y D respectivamente tienen en común 1633 Hz como su tono alto. En estos días, estas teclas de función son empleados principalmente en aplicaciones especiales tales como repetidores de radioaficionados para sus protocolos de comunicación, los módem y circuitos de tonos al tacto (*touch tone*) también tienen tendencia a incluir los pares de tonos A, B, C, y D. Estos no han sido usados para el servicio público en general.

El esquema de marcado DTMF fue diseñado por los *laboratorios BELL* e introducido a los Estados Unidos a mediados de los años 60 como una alternativa para a la marcación por pulsos o rotatoria. Ofreciendo incremento en la velocidad de marcado, mejorando la fiabilidad y la conveniencia de señalización de punto a punto. Muchas aplicaciones en las telecomunicaciones

requieren de transmisión de señales DTMF para el envío de datos y marcado. El estándar DTMF fue diseñado originalmente por los Laboratorios Bell para su uso en los sistemas telefónicos de AT&T.

En conclusión, DTMF es el sistema de señales usado en los teléfonos para el marcado por tonos, estos son el resultado de la suma algebraica en tiempo real de dos señales sinusoidales de diferentes frecuencias. Los tonos altos se diseñaron aproximadamente 2 dB mayores en amplitud respecto a los tonos bajos para compensar las pérdidas en las líneas de transmisión y conexión con las centrales telefónicas. La figura 4.13 muestra una gráfica en el dominio de la frecuencia de los tonos que componen las señales DTMF.

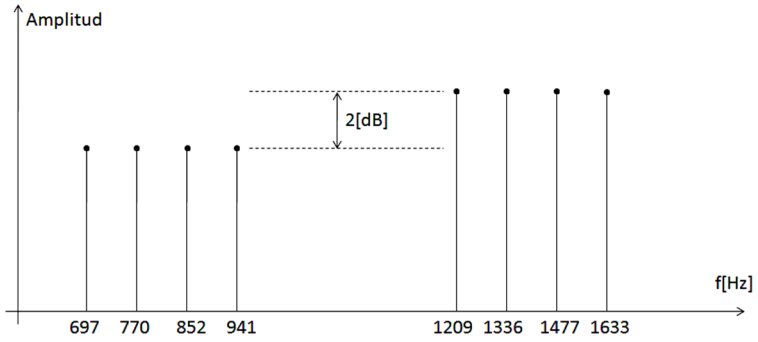


Figura 4.13. Gráfica en frecuencia de los tonos que componen el sistema DTMF.

4.4.4. ALGORITMO DE DETECCIÓN DE SEÑALES DTMF.

El procedimiento a seguir para la detección de las señales DTMF presentes en el audio del teléfono local del sistema de seguridad incorpora todos los conceptos teóricos detallados con anterioridad. Puesto que las señales de audio presentes son continuas, forzosamente se deben pasar por un proceso de muestreo para obtener unas señales discretas que puedan manejarse de manera digital. Una vez que se ha obtenido una señal discreta y acotada a cierta cantidad de muestras, se procesarán dichos datos con el objetivo de obtener la DFT de la señal muestreada en cuestión para que finalmente por medio de un algoritmo de obtención de máximos se encuentre la posición de las dos espigas que caracterizan a la señal DTMF que se haya ingresado y por ende la tecla que se presionó para realizar dicha labor. La figura 4.14 muestra el diagrama que explica los pasos a seguir para el proceso total de detección de tonos DTMF.

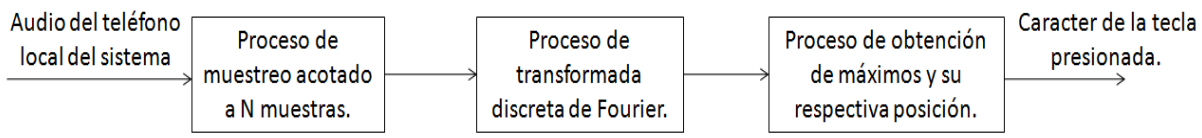


Figura 4.14. Diagrama de bloques del proceso de detección de los tonos DTMF y sus respectivo caracteres.

Para llevar a cabo todo el proceso de detección descrito anteriormente se deben involucrar tanto la PC como el microcontrolador y su comunicación debe ser lo más rápida y eficiente posible ya que la pérdida de datos en este proceso es un lujo que no puede darse al sistema. Puesto que el PIC18F4550 posee un convertidor A/D con doce canales y ya se ha descrito la facilidad de manejo de dicho módulo, este dispositivo será el encargado de llevar a cabo el proceso de muestreo antes detallado. Para ello el microcontrolador recopilará una cantidad de *N=500* muestras con un periodo de muestreo *T=60 µseg*, es decir con una frecuencia de *f=16.67 KHz* tal como lo muestra la figura 4.15.

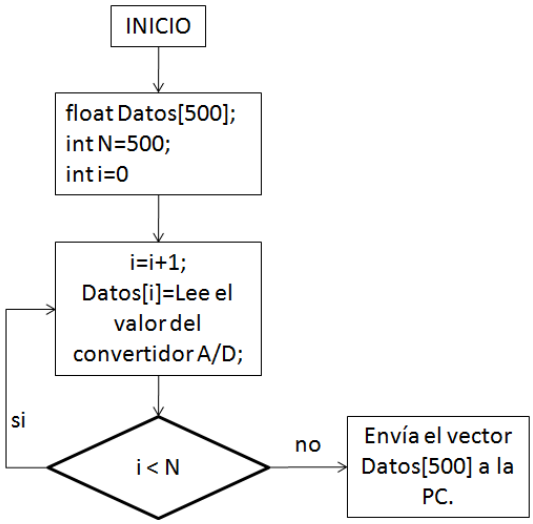


Figura 4.15. Diagrama de flujo del algoritmo de muestreo.

Un código en compilador *PIC C* de *CCS* que podría ejecutar la labor descrita por el diagrama 4.15 resultaría de esta manera:

```
void main(void)                    // Comienzo del programa principal.
{
char datos[500];                   // Se declara el vector donde se guardarán los datos
                                   // resultantes del muestreo.
int16 cont;                        // Se declara la variable que servirá de conteo para el ciclo.
setup_adc(ADC_CLOCK_INTERNAL);     // Se da de alta y configura el convertidor A/D del microcontrolador.
setup_adc_ports(ALL_ANALOG);
set_adc_channel(0);
delay_us(10);
cont=0;                            // Se inicializa el contador en cero.
      do                           // Se comienza el ciclo.
      {
        datos[cont]=read_adc();    // Se lee el valor del puerto A/D y se guarda en el vector Datos con índice
                                   // cont.
        cont++;                    // Se incrementa al valor del contador.
        delay_us(60);              // Se da un retardo de 60 useg como periodo de muestreo.
      }
     while(cont<500);              // Cuando el contador llega a 500 se interrumpe el muestreo.
```

```
//
// Espacio reservado para el código de envío del vector Datos[] hacia la PC…
//
 }
```

En el código anterior no se encuentra descrito el encabezado, el proceso de envío de los datos hacia la PC, ni las archivos de biblioteca necesarios para llevar a cabo dicha función debido a que este es un tema que por su importancia será tratado hasta el capítulo 5 cuando se detallen en concreto los métodos de implementación para el sistema de seguridad.

Una vez que el microcontrolador ha tomado un total de *N=500* muestras y las ha enviado satisfactoriamente a la PC, por medio de un protocolo seguro de comunicación que se describirá en el capítulo 5, el equipo de cómputo por medio de la aplicación desarrollada en *Visual Basic 6.0* aplicará el algoritmo descrito por el diagrama de la figura 4.16 en donde se puede observar los bloques conformados por las ecuaciones 4.12, 4.13 y 4.14.

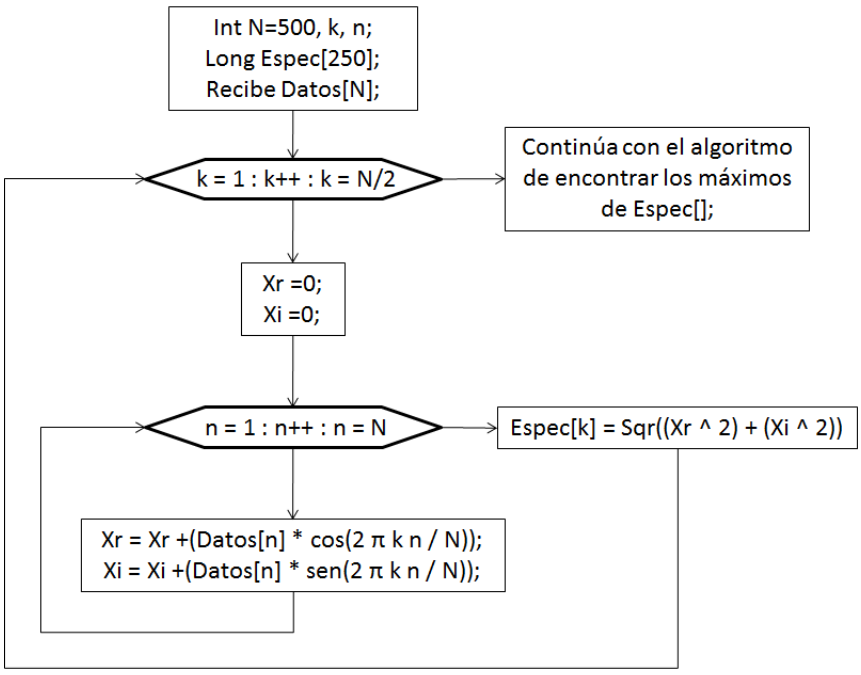


Figura 4.16. Diagrama de flujo de la transformada discreta de Fourier.

En el diagrama de la figura 4.16 se está trabajando con el vector *Datos[]* mismos que se reciben del proceso de muestreo, provenientes en este caso del microcontrolador. Se trabaja con la misma variable *N* que representa la cantidad de muestras tomadas. Las variables *k* y *n* son índices que serán empleados durante el proceso como control de los ciclos. El ciclo controlado por la variable *k* se incrementa únicamente hasta *k=N/2* para aprovechar la propiedad de simetría de la DFT. No se agregaron los bloques del algoritmo de reconocimiento de máximos con el objeto de detallar únicamente el proceso de realización de la DFT. Además

dicho algoritmo es relativamente sencillo y su justificación se detallará en el capítulo 6 debido a que se toman en cuenta resultados analizados con ayuda de MATLAB.

El siguiente código editado en Visual Basic 6.0 realiza el algoritmo descrito por la figura 4.16, no se incluyen la declaración de las variables, ni el proceso de recepción segura de los datos provenientes del microcontrolador, el cual que será explicado en el capítulo 5 de este texto.

```
For k = 0 To 249                'Proceso de Transformada discreta de Fourier.
    Xr = 0
    Xi = 0
    For n = 0 To 499
      Xr = Xr + (Datos(n) * Cos(2 * 3.1415926536 * k * n / 500))
      Xi = Xi + (Datos(n) * Sin(2 * 3.1415926536 * k * n / 500))
    Next n
    Espec(k) = Sqr((Xr ^ 2) + (Xi ^ 2)))
Next k
```

Finalmente mediante la ejecución de un algoritmo de detección de máximos con respecto a un vector de datos, se podrá encontrar la posición exacta de las dos espigas que caracterizan a los tonos DTMF, y por ende el reconocimiento de las teclas telefónicas que hayan sido presionadas. Como ya se mencionó, este algoritmo será descrito en el capítulo 5 y su justificación se encuentra expuesta con base en los resultados mostrados en el capítulo 6, en donde básicamente se comparan y grafican los procesos realizados a unas señales ideales DTMF con los obtenidos y realizados durante el muestreo y procesamiento del audio telefónico del sistema.

4.5. PERIFÉRICOS DE SALIDA.

Los periféricos de salida componen el último módulo del sistema, éste puede estar conformado por diversos elementos o dispositivos, la cantidad de elementos y capacidad de los mismos dependerá completamente del tipo de sistema de seguridad que se posea y las necesidades que el usuario tenga. A continuación se especifican los periféricos de salida que se pretende tener en el sistema de seguridad desarrollado en este proyecto:

1. Luces y alarma sonora.
2. Comunicación telefónica.
3. Intercomunicador o altavoz.
4. Chapa electrónica.

4.5.1. LUCES Y ALARMA SONORA.

La respuesta clásica o más conocida de la mayoría de los sistemas de seguridad para casa habitación y negocios particulares es la activación de luces destellantes y una sirena o alarma sonora, cuyo objetivo principal es el llamar la atención de las personas que se encuentren en la zona aledaña a la propiedad. Si bien, esto no impide que algún posible robo u amenaza que se esté llevando a cabo se elimine o desaparezca, sí favorece a que la participación de terceros alerte a las autoridades correspondientes e introduce presión al posible agresor para que se mantenga dentro de la propiedad el menor tiempo posible.

Una forma de desarrollar estos periféricos de salida es acoplando cierta cantidad de focos y timbres comerciales que trabajen con el sistema de energía eléctrica de la propiedad, o de ser posible, con la de una fuente de energía de emergencia que pueda ser implementada en caso de que algún agresor interrumpiera el suministro regular. La activación de estos periféricos será controlada por un bit del puerto de salida del microcontrolador que será acoplado a un relevador que permita dicha activación y aísle la etapa de control de la etapa de potencia. La figura 4.17 muestra un posible diagrama de conexión para un planteamiento como el expuesto anteriormente con únicamente un foco y un timbre. La conexión de las terminales del relevador para la señal de control no está especificada dado que se puede trabajar con lógica positiva o negativa.

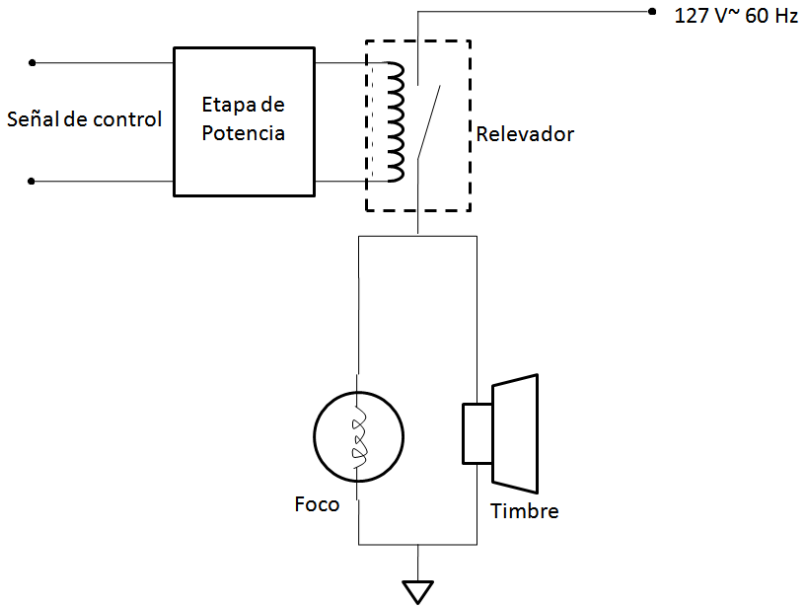


Figura 4.17. Diagrama para una posible conexión de luz y timbre de alarma.

Existen dispositivos que integran ambos elementos (luz y alarma sonora) al mismo tiempo y que son diseñados especialmente para el propósito requerido por un sistema de seguridad. Estos dispositivos pueden ser conectados de una manera similar a la de los periféricos descritos anteriormente.

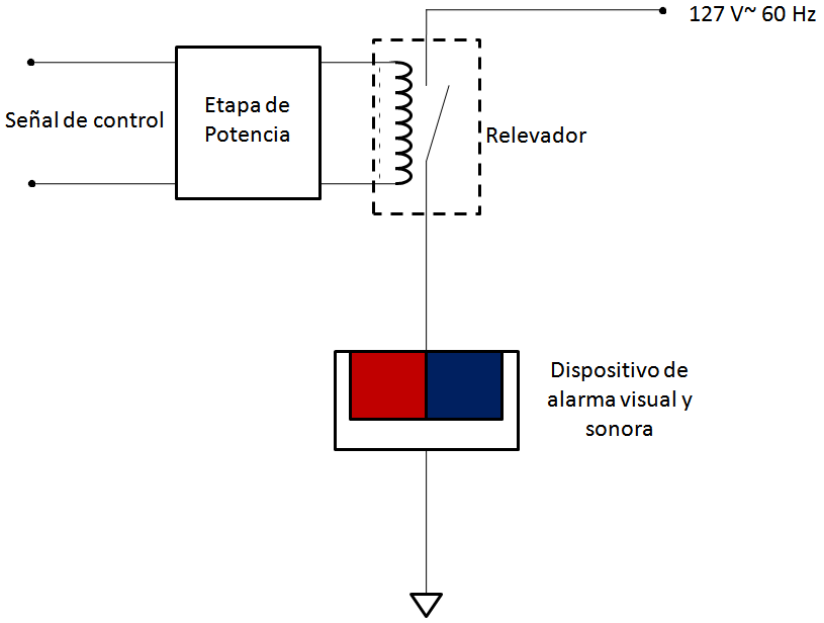


Figura 4.18. Diagrama de una posible conexión de un dispositivo de alarma audio-visual.

4.5.2. COMUNICACIÓN TELEFÓNICA.

La comunicación telefónica como periférico de salida puede permitir una mejor respuesta del sistema en contra de las amenazas que se le presenten, ya que puede informar de manera rápida a las autoridades correspondientes según el evento que se esté presentando, y por lo tanto, ya existe una posibilidad de evitar o combatir la agresión o amenaza que se esté llevando a cabo dentro de la propiedad. Además algo que es importante mencionar con respecto al sistema de seguridad desarrollado en este proyecto, es que la comunicación telefónica se realiza de forma inalámbrica, es decir por medio de un teléfono celular, lo que impide que el agresor pueda impedir o cortar la línea de comunicación desde el exterior de la propiedad, lo que sí es posible con un sistema de comunicación telefónica convencional. Lo descrito anteriormente obliga al posible agresor a llegar hasta la ubicación del teléfono móvil para interrumpir la llamada telefónica que se esté llevando a cabo.

La conexión entre el teléfono celular y la PC se realizará mediante el puerto Bluetooth de ambos dispositivos, la configuración de dicha conexión se detallará dentro del capítulo 5, mientras que la sintaxis de envío de comandos AT hacia el teléfono celular se describe en el capítulo 3 de este texto.

4.5.3. INTERCOMUNICADOR O ALTAVOZ.

El intercomunicador remoto o altavoz del sistema funge como un accesorio del mismo. El objetivo de este periférico es permitir al usuario comunicarse con la persona que se encuentre en el exterior y que haya accionado el timbre de la propiedad, aunque el usuario no se encuentre dentro de la misma.

Para conseguir este objetivo, el sistema debe realizar automáticamente una llamada al usuario usando un cierto código o protocolo de comunicación que el usuario debe conocer, puesto que el sistema debe detectar primero que el usuario ha contestado la llamada para continuar con el procedimiento. Una vez que el sistema ha detectado que el usuario contestó la llamada reproducirá un menú auditivo con el propósito de que el usuario decida si desea o no activar el intercomunicador del sistema para atender a la persona en el exterior. Si bien el diagrama de la figura 4.19 puede verse como parte del módulo de control del sistema, su objetivo es detallar el funcionamiento lógico del altavoz como periférico de salida.

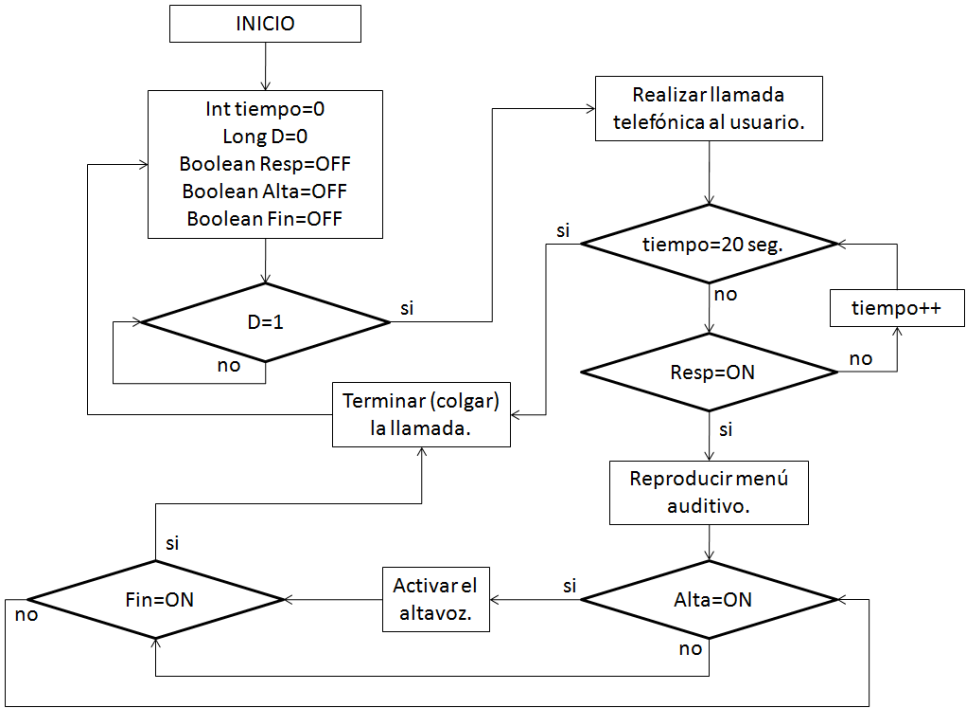


Figura 4.19. Diagrama de flujo del proceso de activación del intercomunicador remoto.

El diagrama de la figura 4.19 básicamente es un algoritmo de toma de decisiones. Este algoritmo es el que se ejecutaría en caso de que el puerto D sea igual a 1. Véanse la figura 4.10 y la tabla 4.2. La variable *tiempo* se incrementaría de forma independiente a la descrita por el algoritmo, y más bien sería dirigida por un control *Timer* de *Visual Basic 6.0* que nos permitiría habilitar la cantidad exacta de segundos antes de terminar automáticamente la llamada, esto con el objeto de no estar marcando indefinidamente al usuario y también poder evadir al buzón de voz.

Las variables *Resp*, *Alta* y *Fin* son una especie de banderas que representan si el *propietario ha respondido a la llamada*, si ha decidido *activar el altavoz* o si ha decidido *terminar con la llamada* respectivamente. En el sistema de seguridad desarrollado, estas variables de tipo booleano son ingresadas vía remota por medio de las señales DTMF que se envían cuando se presiona alguna de las teclas del teléfono particular del usuario. Por lo tanto se puede asignar a cada una de estas acciones o variables una tecla que corresponda a su activación, de tal manera que:

Resp – ***Tecla 1*** – Indica al sistema que el usuario ha contestado la llamada.

Alta – ***Tecla 2*** – Activar/Desactivar el altavoz (toggle).

Fin – ***Tecla 3*** – Indica que el usuario desea terminar con la llamada.

La activación y desactivación del altavoz puede controlarse mediante otro de los pines del puerto de salida del microcontrolador. La figura 4.20 muestra un diagrama general del funcionamiento del altavoz, en el capítulo 5 se detallarán más a fondo los componentes del circuito y las conexiones del mismo.

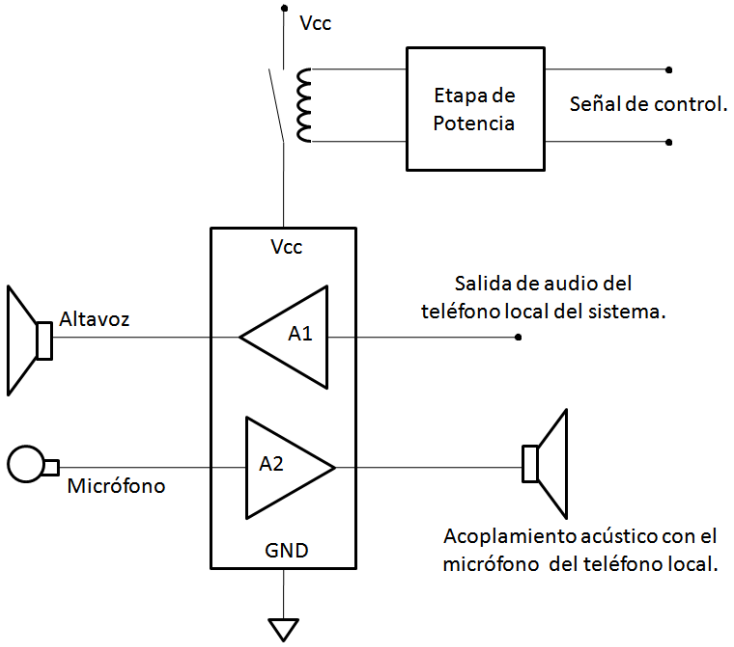


Figura 4.20. Diagrama general del intercomunicador remoto.

4.5.4. CHAPA ELECTROMAGNÉTICA.

Las chapas electrónicas y electromagnéticas se han convertido en dispositivos de bastante popularidad, hoy en día existen no solo casas, sino también cajas fuertes, vehículos y otros artefactos que incluyen estos elementos dentro de sus sistemas, por lo que su uso se vuelve cada día más frecuente. Además libera a los usuarios de la implementación de llaves u otros instrumentos mecánicos para la apertura de los accesos, brindando de esta forma un cierto nivel de automatización que puede variar según el sistema que lo esté llevando a cabo.

Para este proyecto se hará uso de un sistema híbrido simple, donde se implementarán los efectos electromagnéticos para producir un efecto mecánico sencillo como respuesta. El sistema que conforma este periférico de salida es controlado por otro de los bits del puerto de salida del microcontrolador. Por medio del estado de este bit de control se habilita o inhabilita la corriente que atraviesa un solenoide, que a su vez genera un campo magnético que interactúa con un perno o eje metálico que posibilita o niega la apertura de la puerta de la propiedad. Para que en el estado ordinario del sistema dicho perno no permita la apertura de la

puerta, este elemento puede posicionarse con ayuda de otro elemento mecánico como puede ser un resorte por ejemplo, por lo que la fuerza de atracción magnética entre el solenoide y el eje metálico debe ser capaz de vencer la oposición ejercida por el resorte o el otro elemento que realice dicha labor. La figura 4.21 muestra un diagrama que ilustra el funcionamiento de este periférico.

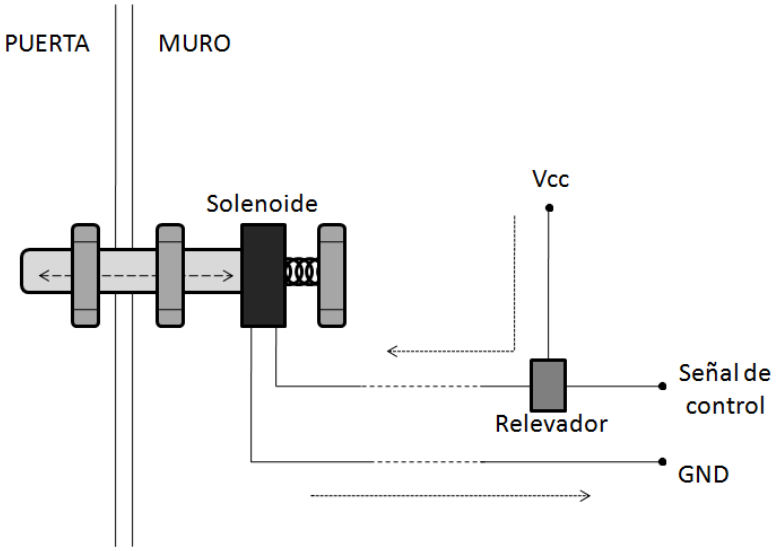


Figura 4.21. Diagrama general de la estructura de la chapa electromagnética.

Para que el usuario pueda activar la chapa electromagnética por medio de la interacción telefónica con el sistema de seguridad, éste último debe ser capaz de responder automáticamente a una llamada del usuario. Dado que el sistema de seguridad trabaja con la detección de señales DTMF, se programará al teléfono celular local con un sonido especial de llamada entrante compuesto por un ciclo de tonos DTMF que se activará únicamente con el número particular del usuario. Una vez que el sistema haya detectado la llamada entrante del usuario y haya respondido automáticamente a dicho evento, reproducirá igual que en casos anteriores un menú auditivo especial con el propósito de ofrecer la activación de la chapa y por ende habilitar la apertura de la puerta. La figura 4.22 muestra el algoritmo de control o toma de decisiones con respecto a la chapa electromagnética.

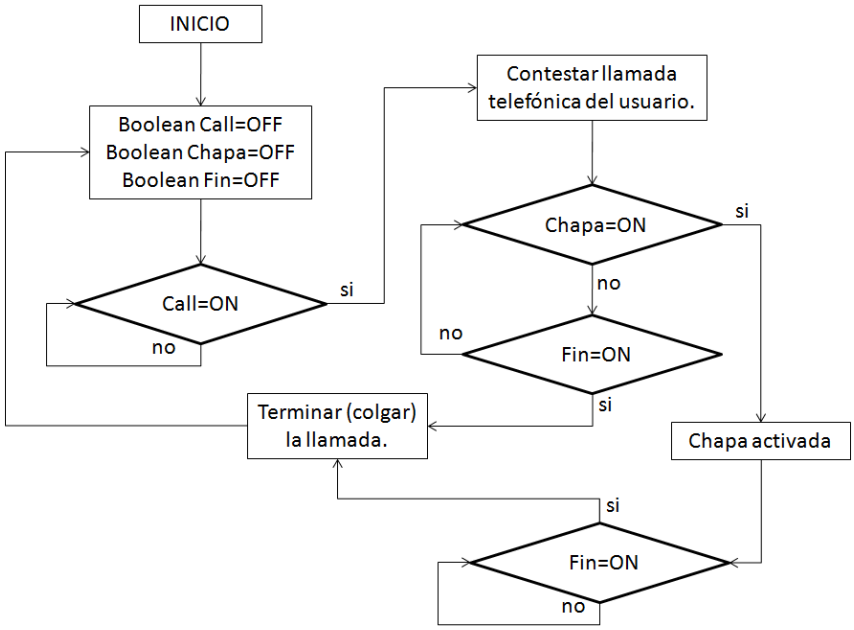


Figura 4.22. Diagrama de flujo del proceso de activación de la chapa electromagnética.

Al igual que en el algoritmo de control del intercomunicador, en éste las variables de tipo booleano *Call*, *Chapa* y *Fin* representan los comando de *contestar llamada*, *activar chapa* y *terminar llamada* respectivamente. Las variables *Chapa* y *Fin* son ingresadas por el usuario presionando las teclas de su teléfono móvil, mientras que la variable *Call* representa la detección del tono especial de aviso de llamada entrante del sistema. Una vez que el usuario decide terminar con la llamada, el sistema desactiva automáticamente la chapa, inhabilitando de esta forma la apertura de la puerta, por lo que el usuario debe decidir terminar con la llamada hasta que haya ingresado a la propiedad y cerrado bien la puerta. Las teclas que corresponden a los dos comandos antes mencionados son:

Fin – ***Tecla 3*** – Termina con la llamada y desactiva la chapa electromagnética.

Chapa – ***Tecla 4*** – Activa/Desactiva la chapa electromagnética (Toggle).

4.6. RESUMEN Y SÍNTESIS.

En este capítulo se detallaron los conceptos teóricos de todos los componentes del sistema de seguridad. Se describieron en un comienzo los componentes del sistema de detección, detalles de los *opto-interruptores* que se pretenden utilizar y un diseño teórico de la manera en que irán conectados. De la misma forma se detallaron las características del microcontrolador *PIC18F4550* y la tarjeta de desarrollo *UPRsys V0308*, se describió el proceso de captación de las señales de los sensores y la manera de comunicarse con la PC.

También se analizó un algoritmo de toma de decisiones general para el módulo de control, conformado en su mayor parte por la PC. Se detalló la teoría básica del procesamiento digital de señales, así como el *teorema del muestreo* y la *transformada discreta de Fourier*, que permiten al sistema la detección de señales DTMF provenientes del audio del teléfono celular. Se analizaron las propiedades de dichas señales y se diseñó un algoritmo de detección con base en las herramientas de la teoría.

Por último se detallaron los periféricos de salida que serán implementados en el sistema, se describió de forma teórica la composición de las *luces y alarma sonora*, el *intercomunicador remoto*, la *comunicación telefónica* y la *chapa electromagnética*. Se diseñaron los algoritmos particulares de toma de decisiones que complementan el algoritmo mostrado para el módulo de control.

De esta manera se pretende mostrar una estructura total prácticamente definida para el sistema de seguridad. Quedan pendientes algunos detalles prácticos que serán descritos en el próximo capítulo y completarán el desarrollo del proyecto, para que de esta manera se pueda llegar al proceso de implementación y poder realizar las pruebas pertinentes que permitan la correcta evaluación del mismo.

CAPÍTULO 5.
IMPLEMENTACIÓN DEL SISTEMA DE SEGURIDAD.

En este capítulo se detallarán los diagramas esquemáticos que representan las conexiones y acoplamientos correspondientes entre los elementos y módulos del sistema. Además, por medio de un análisis con base en los algoritmos de toma de decisiones se establecerá un diagrama de estados del sistema que permitirá el correcto funcionamiento del módulo de control.

Otra cuestión importante dentro de este capítulo será el diseño de un protocolo de comunicación entre el microcontrolador y la PC que asegure la correcta transmisión de datos, recordando que dicho proceso se realiza de manera asíncrona y dando prioridad a comunicar algunos eventos, entre los cuales existen algunos en que es permitido no captar datos que no resulten de vital importancia para el sistema.

De igual forma se detallará la configuración necesaria de la conexión Bluetooth del teléfono celular con la PC, para que después dicho dispositivo sea reconocido como modem y se pueda entablar comunicación de tipo serie mediante comandos AT que permitan el control telefónico necesario para el sistema de seguridad desarrollado en este proyecto.

5.1. MÓDULO DE DETECCIÓN.

El sistema dispone de los sensores que detectan los diversos eventos o amenazas que son de interés para el mismo, se ha especificado y descrito la naturaleza de los sensores que pretenden utilizar en este proyecto, y en adelante se describirán los diagramas esquemáticos que presentan la conexión de dichos elementos con el módulo de control del sistema.

5.1.1. CONEXIÓN DE LOS SENSORES Y EL TIMBRE.

En la figura 5.1 se muestra el diagrama esquemático que muestra la conexión de los sensores con el microcontrolador, además de la conexión del timbre:

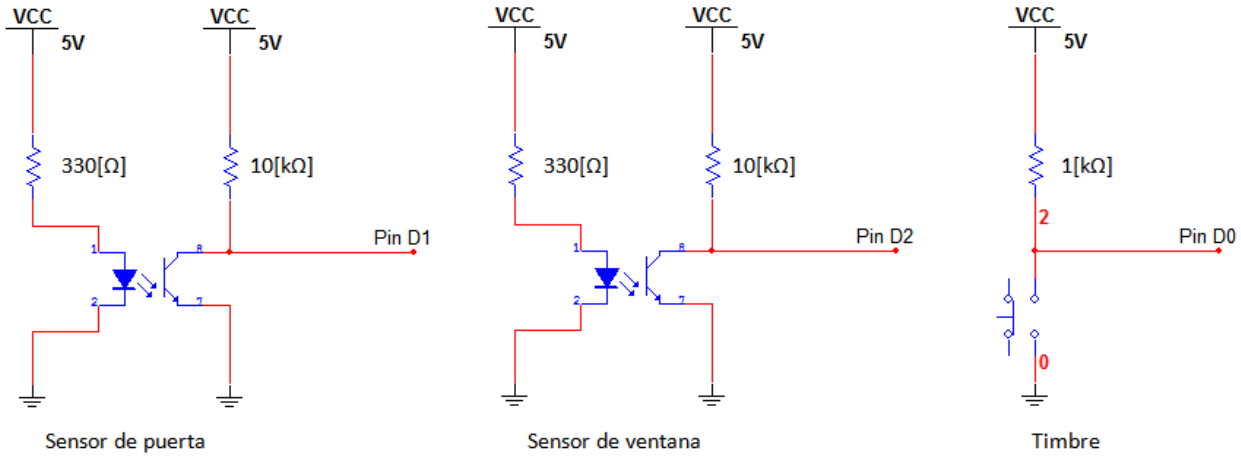


Figura 5.1. Diagrama esquemático de la conexión del timbre y los sensores con el microcontrolador.

En la figura 5.1 se observa que el sistema está configurado para funcionar con lógica negativa. La razón de esto es que además de simplificar la utilización de los sensores por medio del acople de los opto-interruptores, permite que el sistema recopile las señales de manera un poco más estable, puesto que resulta más sencillo fijar una señal al voltaje de referencia GND que a 5V.

Tanto los sensores como el timbre no se conectan de forma inalámbrica con el microcontrolador, por lo tanto los cables de conexión resultan una importante fuente de ruido para los componentes involucrados. Por esta misma razón es que el timbre de la propiedad también se encuentra configurado para funcionar con lógica negativa.

Por lo tanto todas las señales provenientes de los sensores y el timbre, permanecerán fluctuando en valores muy cercanos a 5V (considerando el factor de ruido antes mencionado), y

caerán a 0V cuando alguno de ellos cambie su estado físico debido a alguno de los eventos que se haya presentado.

5.2. INTERFAZ DEL MICROCONTROLADOR.

Dentro del sistema de seguridad desarrollado, el microcontrolador actúa como interfaz entre los sensores, el audio telefónico y el equipo de cómputo. Dadas las características del microcontrolador, no es posible que el propio microcontrolador realice todas las labores correspondientes al procesamiento de las señales, por lo que se limita a adecuar y recopilar las mismas para posteriormente comunicarlas hacia la PC.

5.2.1. ALGORITMO DE TRABAJO DEL MICROCONTROLADOR.

El microcontrolador recopila las señales de los sensores y el audio telefónico, sin embargo, dichas señales son de diferente naturaleza, es decir, las señales provenientes de los sensores son recibidas de forma digital y se conectan a un puerto configurado para dicho tipo de señales, mientras que la señal de audio telefónico es analógica y tiene que ser recibida por un puerto configurado como convertidor analógico-digital.

Por esta razón el algoritmo de trabajo del microcontrolador realiza una detección con jerarquía de las señales. Da prioridad a la detección y comunicación de señales DTMF, y el resto del tiempo lo dedica a la recopilación y comunicación de las señales de los sensores. La figura 5.2 muestra el diagrama a bloques que describe el algoritmo de operación programado para el PIC18F4550.

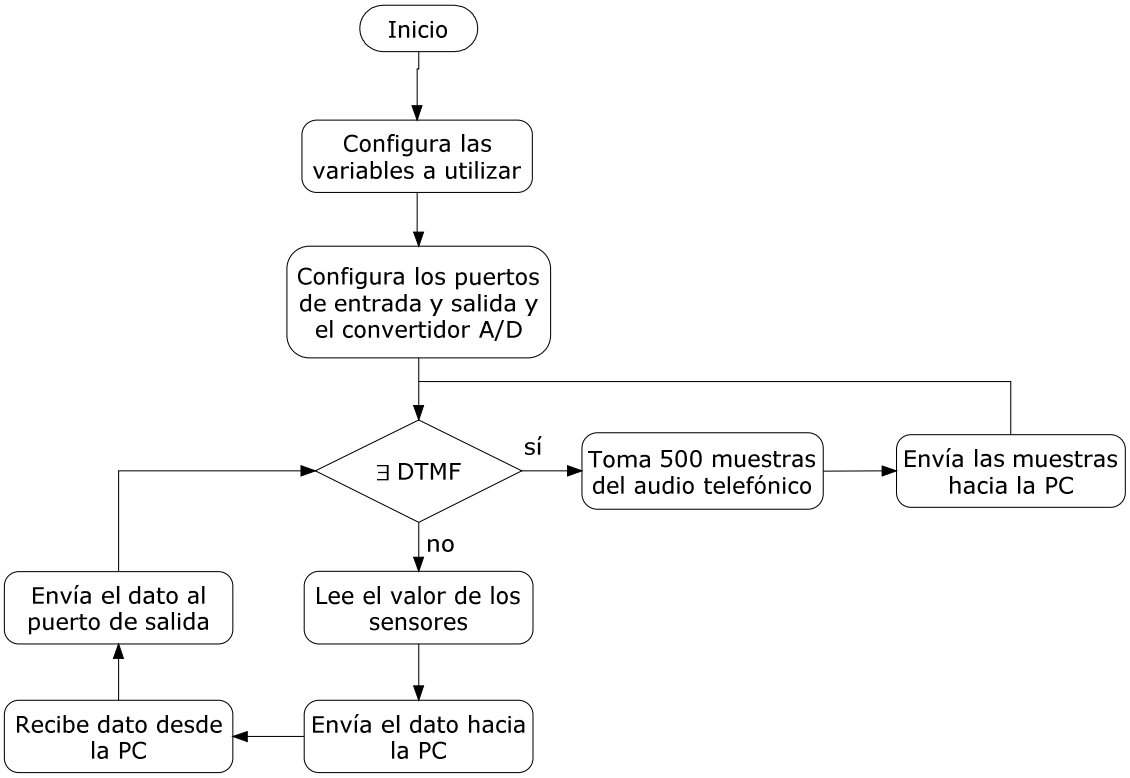


Figura 5.2. Algoritmo de operación del microcontrolador.

Para poder detectar la presencia de las señales DTMF en el audio telefónico, sin la necesidad de estar realizando el proceso de muestreo de audio periódicamente, es necesario aplicar un algoritmo especializado que se encargue únicamente de detectar el momento en que exista una señal de audio presente. Dicho algoritmo debe tomar en cuenta el ruido que se presenta en la línea. Un sub-diagrama que correspondería al bloque de decisión "∃ DTMF" (existe DTMF) se muestra en la figura 5.3, y aquel que describe el algoritmo de trabajo del microcontrolador se muestra en la figura 5.2.

Como puede observarse en la figura 5.3, es necesario programar un intervalo específico, que cubra sin problemas la amplitud máxima que pueda tener la señal de ruido presente. La figura 5.4 muestra una gráfica de la amplitud de la señal de ruido presente en la línea del audio telefónico. Cuando el voltaje en la línea telefónica rebasa los límites establecidos anteriormente significa que existe una señal de audio presente, y para el propósito de este proyecto esta señal debe corresponder a un tono DTMF.

Otra cuestión importante del diagrama de la figura 5.2, es que el microcontrolador una vez activado entra en un ciclo de trabajo que únicamente es finalizado cuando todo el sistema se desactiva. El microcontrolador toma sólo una muestra de la línea de audio telefónico, no precisamente de forma periódica, y conforme el valor de ésta puede tomar la decisión de

comenzar el muestreo de audio y su posterior transmisión, o bien, recopila el valor de los sensores y los comunica a la PC.

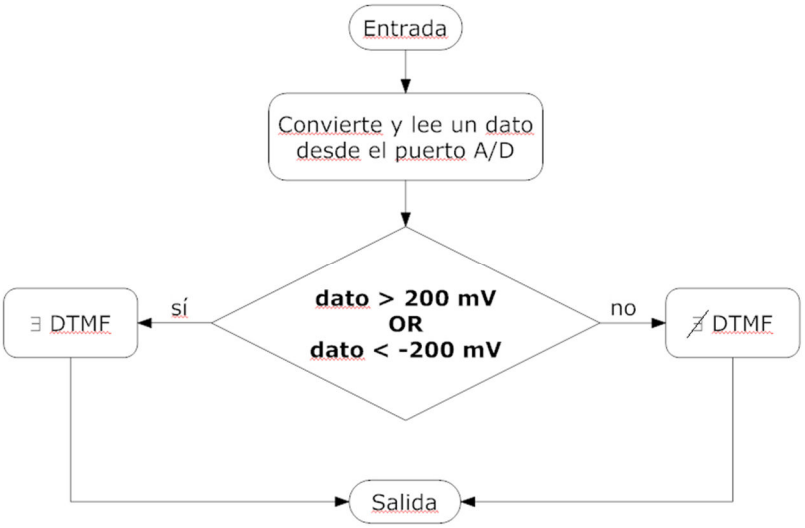


Figura 5.3. Detección de audio en la línea telefónica.

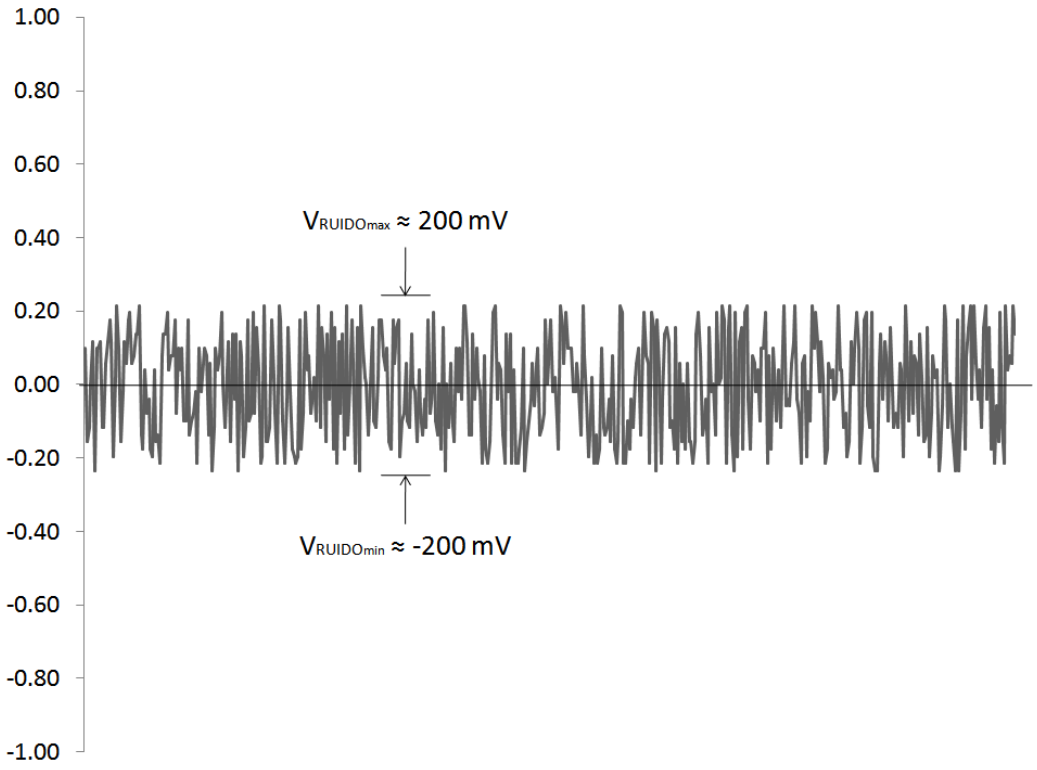


Figura 5.4. Gráfica del ruido presente en la línea de audio telefónico.

5.2.2. CONEXIÓN DEL AUDIO DEL TELÉFONO MÓVIL Y MUESTREO.

Para llevar a cabo la conexión del audio telefónico al microcontrolador es necesario implementar un pequeño circuito de acoplamiento que facilite la conversión de la señal y haga más eficiente la recopilación de los datos.

El primer punto a considerar para el diseño de este circuito, es que el convertidor A/D del microcontrolador PIC18F4550 es capaz de trabajar en un intervalo de voltajes que va de 0V hasta 5V. Sin embargo, la señal proveniente del audio telefónico es aleatoria y varía entre voltajes positivos y negativos, por lo tanto es necesario acoplar un voltaje de *offset* junto con la señal de audio a la entrada del microcontrolador, de esta manera se obtendrá en dicha entrada una señal proporcional al audio telefónico pero que varía únicamente entre valores de voltaje positivos.

Otra cuestión importante que es necesario considerar, es la amplitud máxima de la señal de audio que puede obtenerse directamente del dispositivo (teléfono celular), ya que puede entrar en conflicto con la resolución del convertidor A/D del microcontrolador y provocar un proceso de muestreo erróneo y menos eficiente. Por lo que es conveniente emplear una etapa de amplificación antes de la conversión analógica-digital.

La figura 5.5 muestra el diagrama esquemático del circuito empleado para acoplar y conectar la salida de audio del teléfono celular al puerto A/D del microcontrolador.

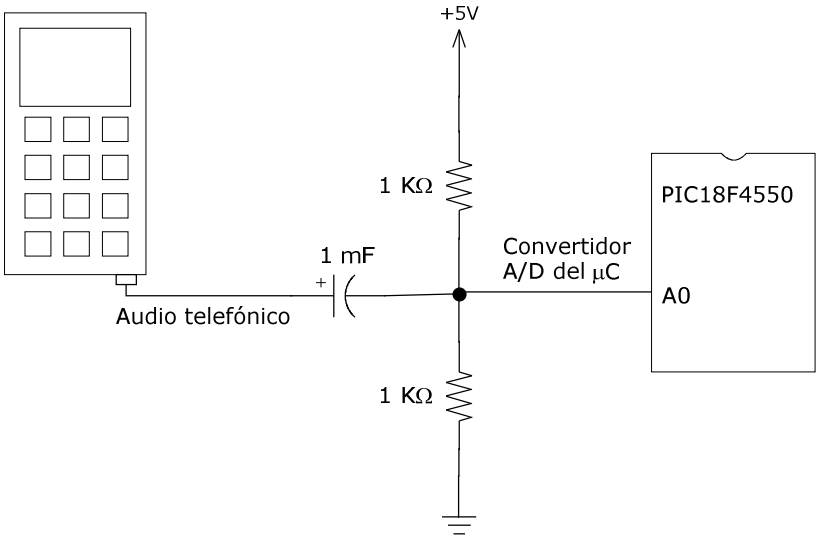


Figura 5.5. Diagrama esquemático del circuito de acoplamiento entre el audio telefónico y el microcontrolador.

Como se muestra en la figura 5.2, una vez que se detecta la presencia de una señal DTMF en la línea de audio del teléfono, se comienza inmediatamente el proceso de muestreo, cuyo algoritmo ya se ha descrito en el capítulo anterior.

El código programado para el proceso de muestreo es el siguiente:

```
comp=read_adc();                  // Se lee el valor del puerto A/D y se almacena en la variable "comp".
if ((comp>140)||(comp<115))   // Si el valor de "comp" es mayor a 140 o menor a 115 debido al Δ(dato)
{                                      // mencionado anteriormente, entonces comienza el proceso de muestreo.
cont=0;                               // Inicializa una variable de conteo.
do                                     // Comienza un ciclo para llevar a cabo el proceso de muestreo.
{
  datos[cont]=read_adc();     // Lee el valor del puerto A/D y se almacena en el arreglo "datos"
cont++;                             // con índice "cont". Se incrementa en una unidad el índice y se programa un
delay_us(40);                      // retardo de 40 microsegundos, para lograr una frecuencia de muestreo de
}                                      // aproximadamente 25 KHz.
while(cont<500);                // Se toma un total de 500 muestras.
```

5.2.3. PROTOCOLO DE COMUNICACIÓN CON LA PC.

Para el correcto funcionamiento del sistema, es muy importante que los datos enviados desde el microcontrolador hacia la PC sean correctamente recibidos, y dado que la comunicación de tipo serie establecida entre ambos dispositivos funciona de manera asíncrona debe existir un protocolo que evite la pérdida de datos de jerarquía mayor (señales DTMF).

Los datos relacionados a las señales DTMF poseen una mayor jerarquía en la comunicación, debido básicamente a que conllevan la información que permite la interacción directa entre el usuario y el sistema de seguridad.

Las señales provenientes de los sensores no cambian o se modifican a una velocidad alta, además de ser de naturaleza digital lo que hace la detección más inmune a ruido, y dado que no se espera que los sensores se estén activando y desactivando de forma frecuente, estas señales pueden acoplarse y adecuarse por hardware para asegurar su correcta comunicación.

El algoritmo de transmisión se encontrará procesando las señales provenientes de los sensores, por lo que carece de sentido utilizar un protocolo de comunicación complejo para el envío y recepción de datos correspondientes a los sensores. En resumen, el protocolo de comunicación utilizado en el sistema, solo se implementa para el procesamiento de las señales DTMF. La figura 5.6 muestra el algoritmo diseñado para llevar a cabo dicha labor.

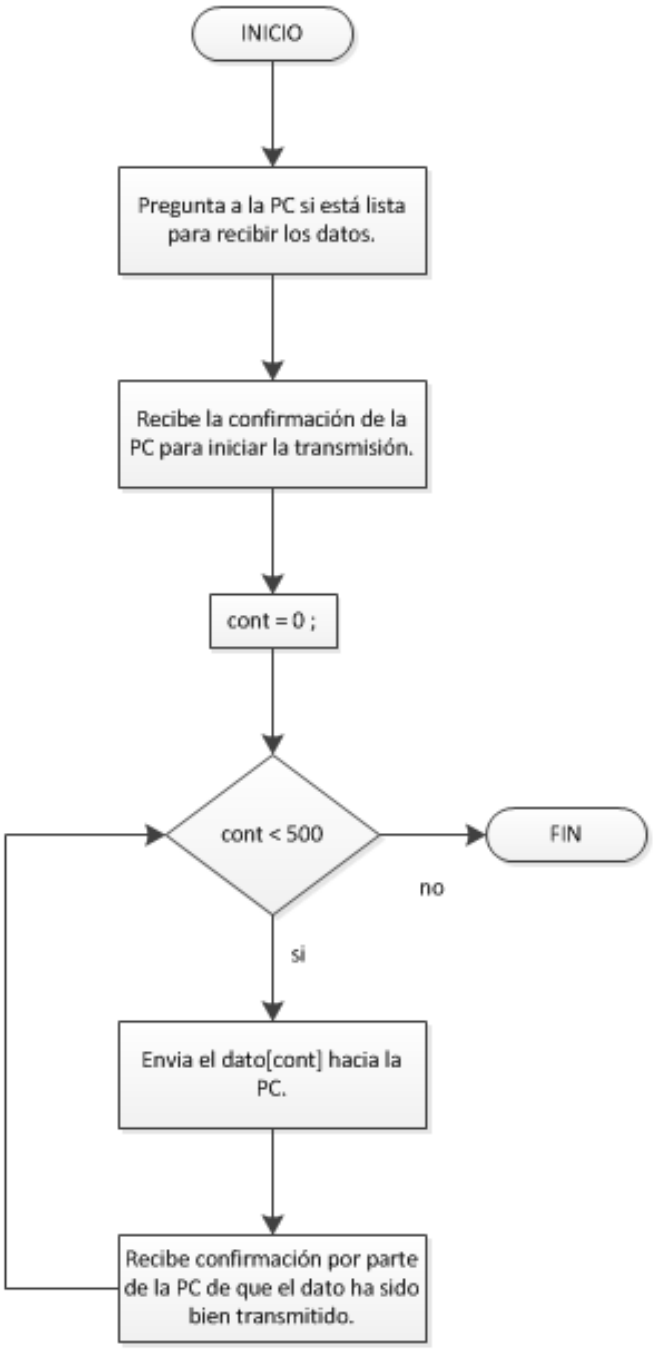


Figura 5.6. Algoritmo de comunicación del microcontrolador con la PC.

El código programado en el microcontrolador para ejecutar este algoritmo se muestra a continuación:

```
do                          //Comienza el proceso de comunicación con la PC.
{
usb_cdc_putc('L');          //Envía el carácter “L” hacia la PC y espera a que sea tomado por la PC.
do
```

```
        {
comp=usb_cdc_getc();                //Recibe un caracter como respuesta, proveniente de la PC.
}
while(usb_cdc_kbhit()==FALSE);    //Asegura que el caracter ha sido tomado por el microcontrolador.
}
while(!(comp=='K'));                //Si el caracter recibido es igual a “K” significa que la PC está lista.
cont=0;                             //Reinicia la variable de conteo.
do                                  //Comienza el proceso de envio de datos hacia la PC.
{
usb_cdc_putc(datos[cont]);          //Envía el dato con índice “cont” hacia la PC.
cont++;                             //Incrementa en una unidad el valor del contador.
do
        {
comp=usb_cdc_getc();                //Recibe un caracter como respuesta, proveniente de la PC.
}
while(usb_cdc_kbhit()==FALSE);    //Asegura que el caracter ha sido tomado por el microcontrolador.
}
while(!(comp=='K'));                //Si el caracter recibido no es igual a “K” significa que aún existen más datos
}                                     por recibir.
```

5.3. CONFIGURACIÓN DE LA CONEXIÓN ENTRE EL TELÉFONO Y LA PC.

La habilidad de mayor importancia para el sistema es la capacidad de establecer una llamada telefónica y procesar la misma de manera controlada. Sin embargo, para llevar a cabo dicha comunicación primero el sistema debe tener la certeza de haber establecido y configurado correctamente la conexión entre el teléfono celular y la PC. Mientras que la conexión puede ser probada y establecida automáticamente por el sistema, la configuración debe llevarse a cabo manualmente.

5.3.1. CONEXIÓN DEL TELÉFONO MOVIL CON LA PC VÍA BLUETOOTH.

La conexión entre un teléfono celular y una PC puede configurarse y llevarse a cabo fácilmente cuando ambos dispositivos están capacitados para llevar a cabo una conexión Bluetooth. Los siguientes pasos a seguir ejemplifican el proceso para dar de alta y configurar la conexión Bluetooth entre la PC y un teléfono celular.

1. Encender o activar la comunicación Bluetooth en ambos dispositivos.

2. Desde el menú Bluetooth de la PC, seleccionar la opción “Agregar un dispositivo”. Aparecerá una ventana mostrando todos los dispositivos activos dentro del perímetro de alcance del Bluetooth de la PC, entonces seleccionamos el dispositivo telefónico deseado.

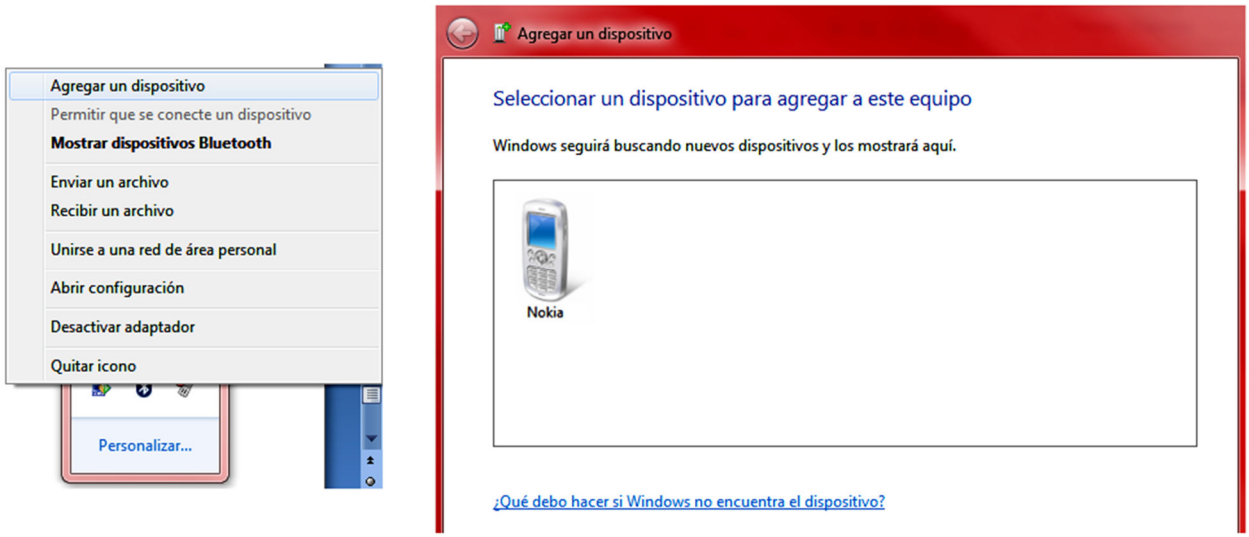


Figura 5.7. Menú para agregar un dispositivo Bluetooth.

3. Si el proceso se realiza satisfactoriamente aparecerá un mensaje de confirmación.

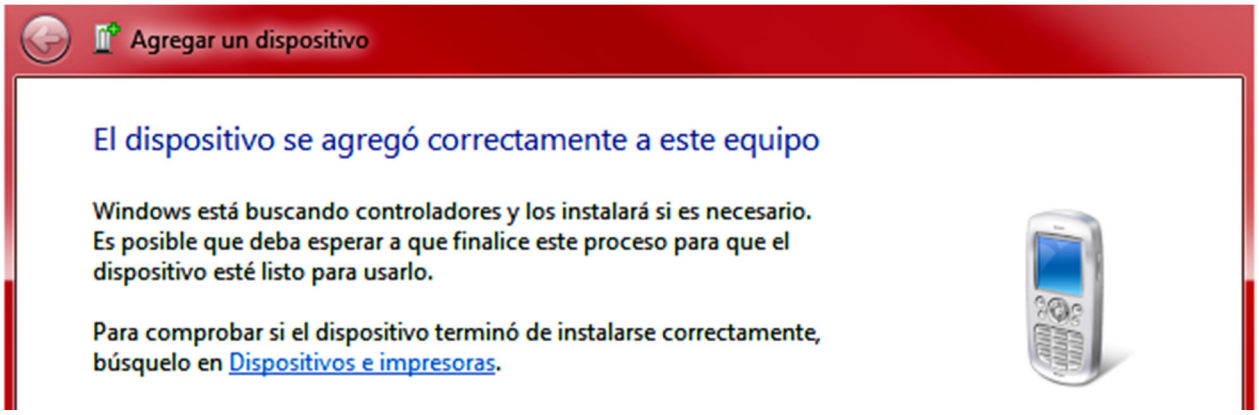


Figura 5.8. El conexión con el dispositivo se configuró correctamente.

La conexión entonces estará lista y en adelante podrá llevarse a cabo automáticamente, siempre y cuando los puertos Bluetooth de ambos dispositivos se encuentren activados. Si la conexión del dispositivo no puede darse de alta correctamente, puede ser debido al número de capacidades de conectividad del teléfono móvil, por lo que podría ser necesario instalar en la PC los controladores o archivos de configuración necesarios para su correcta instalación, por medio de algún CD de instalación o descargándolos directamente desde internet.

5.3.2. CONFIGURACIÓN DEL TELÉFONO MOVIL COMO MODEM.

Antes de concluir que la conexión del teléfono no se ha dado de alta correctamente, es necesario tomar en cuenta que el proceso puede mandar un mensaje de error, solamente por el hecho de que alguna de sus funcionalidades no pudieron darse de alta de forma compatible para ambos dispositivos. Sin embargo, la funcionalidad del teléfono para ser utilizado como modem y establecer una conexión de tipo serie, rara vez causa complicaciones, por lo que solo queda conocer como ha quedado configurada dicha funcionalidad.

Para ello es necesario acceder al menú de "Dispositivos e Impresoras" en Windows, allí se encontrará el dispositivo telefónico dado de alta para la conexión. La figura 5.9 muestra el menú descrito.

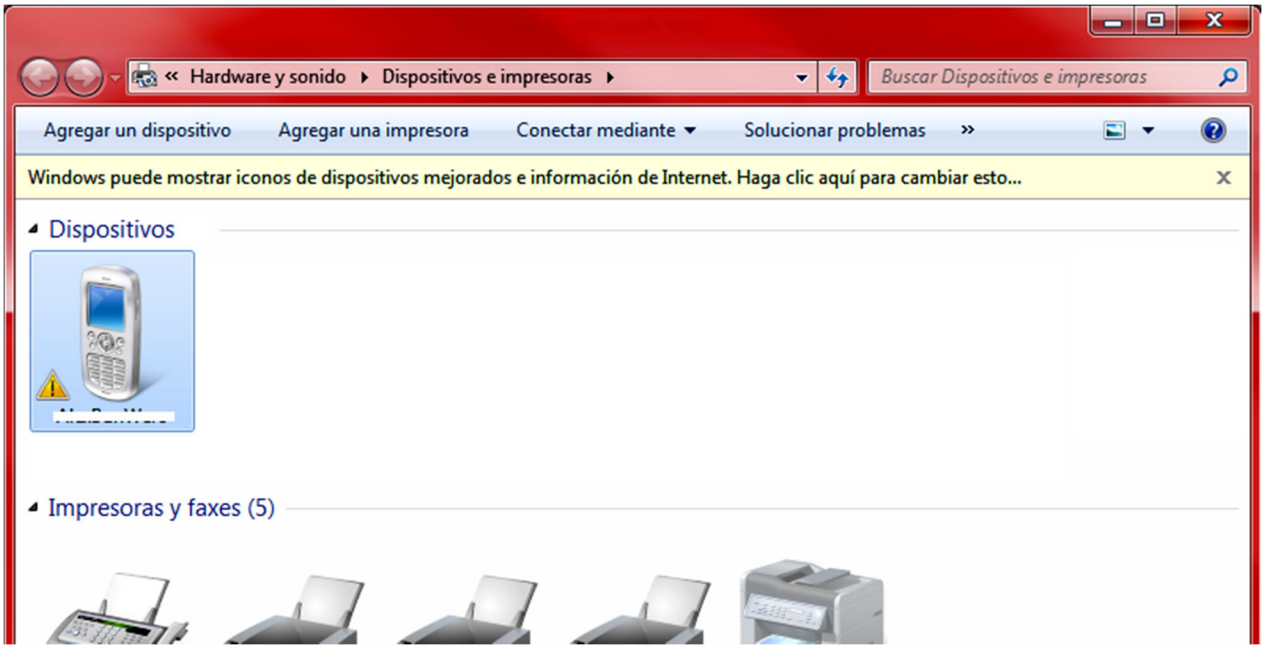


Figura 5.9. Menú Dispositivos e Impresoras de Windows 7.

Si se da un clic con el botón derecho del mouse sobre el dispositivo, y luego se selecciona la opción de *Propiedades* se desplegará una ventana que posee la descripción de las capacidades de conexión del teléfono y aún más importante para el propósito del proyecto, describe el número de puerto COM a utilizar y los detalles de cómo está configurado.

De esta manera el sistema de control podrá dirigir los comandos AT a través del puerto Bluetooth, configurando la conexión al puerto COM indicado. La figura 5.10 muestra una imagen de la ventana descrita anteriormente.

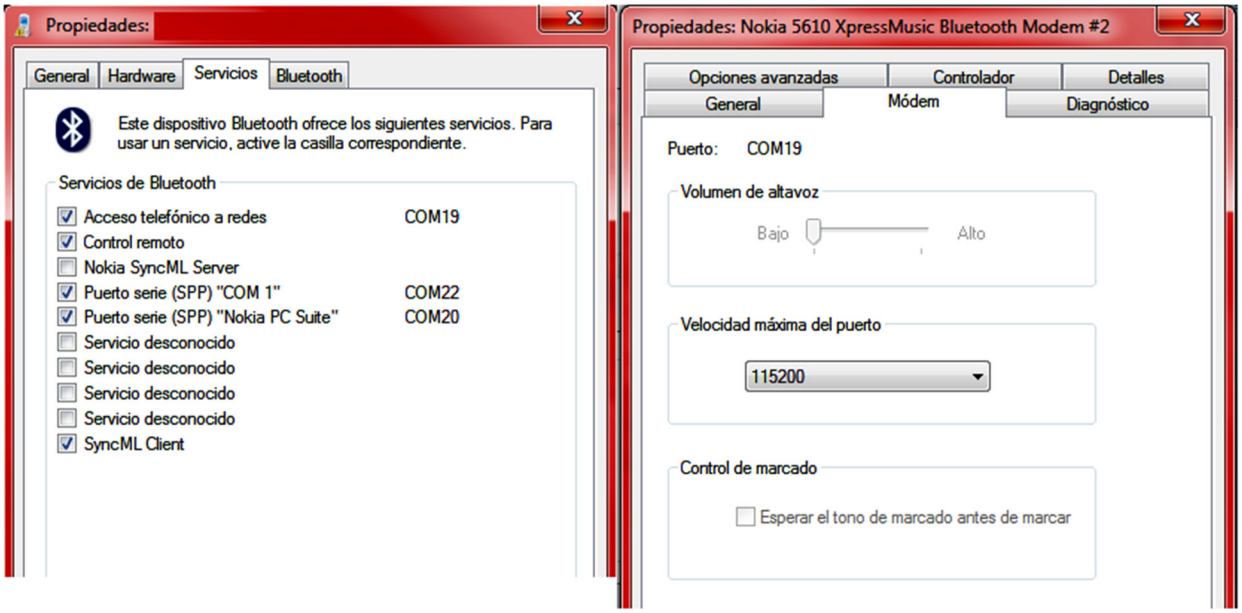


Figura 5.10. Propiedades del dispositivo telefónico.

5.4. MÓDULO DE CONTROL.

El bloque de control del sistema de seguridad recae en la programación de la PC, éste conlleva el procesamiento de los datos de audio, el establecimiento de la llamada telefónica, la comunicación con el microcontrolador y el control sobre la activación y desactivación de los periféricos de salida, así como la respuesta al cambio en los estados de los sensores.

5.4.1. ALGORITMO DE CONTROL POR ESTADOS DEL SISTEMA.

Tal como lo haría un sistema secuencial, el módulo de control responde y toma decisiones asignando un valor único a una variable de tipo entero que representa un estado peculiar del mismo, es decir que todos los eventos que pueden llegar a presentarse se han tomado en cuenta de forma predeterminada y existe una respuesta del sistema previamente preparada para todos y cada uno de los sucesos posibles.

Así cuando el sistema comienza su funcionamiento, puede decirse que éste se encuentra en estado inicial (*E=0*) que significa que ningún suceso se ha llevado a cabo. Todas las entradas posibles como la señal de los sensores, el timbre o el audio telefónico no han cambiado su valor de arranque. Sin embargo, cuando uno de estos eventos sucede y el sistema lo detecta, éste automáticamente asigna un valor de estado diferente a sí mismo y con base en ello toma decisiones, y genera respuestas que a su vez pueden llevar al sistema a desencadenar una serie

de cambios de estado posteriores, hasta que estos mismos lo lleven a entrar en su estado de estabilidad inicial (E=0) nuevamente.

La figura 5.11 muestra el diagrama de estados incluido en la programación de la PC para la toma de decisiones.

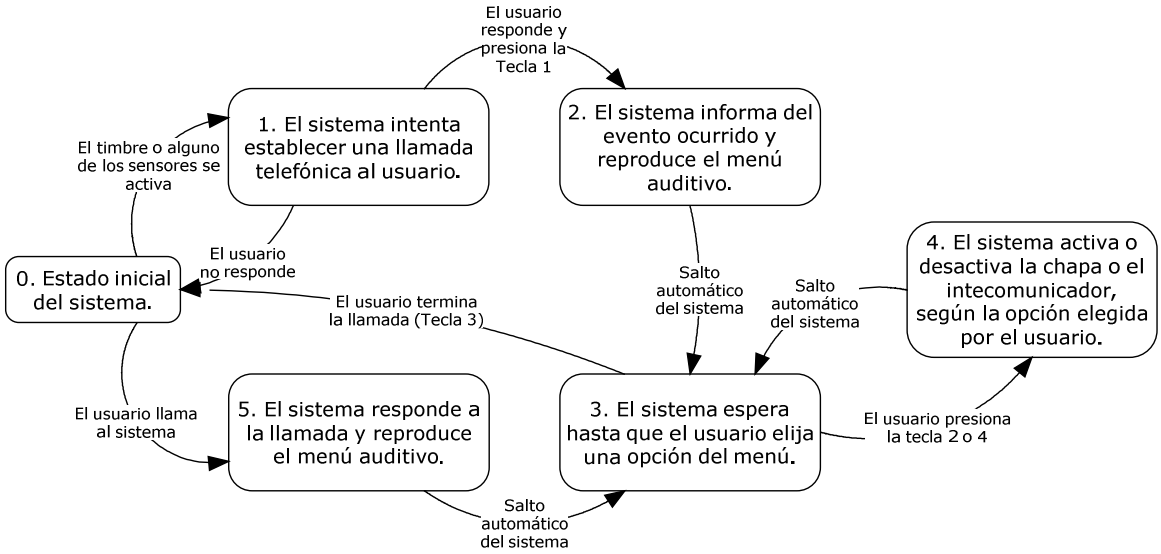


Figura 5.11. Diagrama de estados del sistema.

5.4.2. ESTABLECIMIENTO DE LA COMUNICACIÓN TELEFÓNICA.

En el capítulo 2 se detallaron ciertas características de la comunicación Bluetooth, entre ellas sus protocolos disponibles, haciendo hincapié en el denominado *RFCOMM* el cual consistía básicamente en emular la comunicación serie por un puerto *RS-232*. En el capítulo 3 se detallaron a grandes rasgos los lenguajes de programación utilizados en este proyecto, se describió de forma general el funcionamiento de uno de los controles disponibles en el lenguaje *Microsoft Visual Basic 6.0* denominado *MSComm*. Este control permite al sistema gestionar los puertos COM de la PC para establecer conexión y permitir la comunicación a través de dichos puertos con diversos dispositivos, como módems y microcontroladores.

A continuación se describen nuevamente los comandos que son empleados para dicho objetivo:

MSComm1.CommPort = 19 *'Se especifica el valor del puerto COM para la conexión con el teléfono.*
MSComm1.Settings = "115200,N,8,1" *'Se configuran las características de comunicación con el mismo.*
MSComm1.InputMode = InputModeText *'Los datos se transmitirán como texto en código ASCII.*
MSComm1.PortOpen = True *'Se abre el puerto indicado y se establece la comunicación con el dispositivo.*

Una vez dada de alta la conexión, las instrucciones importantes para enviar comandos AT a un teléfono celular, en el caso de aquellos desarrollados por Nokia Inc., son:

MSComm1.Output = "ATDT 555-1234567;"+vbCr *'Se envía el comando de marcado por tonos al número indicado.*
MSComm1.Output = "ATA;"+vbCr *'Se envía el comando para contestar a una llamada entrante.*
MSComm1.Output = "AT+CHUP;"+vbCr *'Se envía el comando para terminar la llamada o la marcación.*

Todos los procesos para la gestión de las llamadas telefónicas por parte de la PC, son llevados a cabo únicamente con la ayuda de las tres instrucciones descritas anteriormente, y los valores de entrada que representan las señales DTMF en el audio del teléfono.

5.4.3. LA COMUNICACIÓN CON EL MICROCONTROLADOR.

La comunicación que se lleva a cabo con el microcontrolador, se gestiona y da de alta por la PC de forma idéntica a la del teléfono celular. Sin embargo, para configurar la conexión el procedimiento es distinto. Como se detalló en los capítulos 2 y 3, mientras que la conexión Bluetooth del teléfono, integra dentro de sus protocolos de comunicación al denominado *RFCOMM* que permite emular un puerto serie *RS-232*, la conexión del microcontrolador se lleva a cabo mediante uno de los puertos *USB* de la PC.

El hecho es que Microsoft Visual Basic 6.0 no integra controles ni comandos para la gestión de puertos USB con sus protocolos originales de comunicación, por lo que se recurre a la configuración de dicho puerto para que funcione emulando un puerto serie de tipo *RS-232*. Para ello la compañía *Microchip, Inc.* incorporó dentro de los archivos de configuración de sus microcontroladores, a la denominada *usb_cdc.h*, el controlador *mchpcdc.inf* y los comandos *usb_cdc_getc()* y *usb_cdc_putc()*.

La función del archivo *usb_cdc.h* y la de sus comandos de transferencia de datos *usb_cdc_getc()* y *usb_cdc_putc()* ya se describieron en capítulos anteriores. El archivo *mchpcdc.inf* básicamente contiene datos para la configuración del puerto *USB* el cual mapea a registros internos del microcontrolador *enumerándolo* y *asignándole una dirección* para llevar a cabo la comunicación con cualquier otro dispositivo de conexión *USB* pero dando de alta la archivo y comandos antes mencionados para emular la comunicación serie *RS-232*. Además provee a la PC de información básica del dispositivo como la compañía a la cual pertenece el dispositivo (idVendor) y la función principal que realiza el mismo.

Este archivo se puede editar para personalizar la información del dispositivo que será adquirida. A continuación se muestra el controlador *mchpcdc.inf* modificado para el propósito de este proyecto y la correcta configuración de la conexión *USB* para funcionar emulando un puerto serie *RS-232*.

```
; Windows 2000, XP, Vista and 7 setup File for CCS CDC demo
; Facultad de Ingeniería, UNAM 2011
[Version]
Signature="$Windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%CCS%
LayoutFile=layout.inf
[Manufacturer]
%CCS%=CCS
[CCS]
%CCS_CDC%=Reader, USB\VID_0461&PID_0033
[Reader_Install.NTx86]
;Windows2000
[DestinationDirs]
DefaultDestDir=12
Reader.NT.Copy=12
[Reader.NT]
Include=mdmcpq.inf
CopyFiles=Reader.NT.Copy
AddReg=Reader.NT.AddReg
[Reader.NT.Copy]
usbser.sys
[Reader.NT.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys
HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"
[Reader.NT.Services]
AddService = usbser, 0x00000002, Service_Inst
[Service_Inst]
DisplayName = %Serial.SvcDesc%
ServiceType = 1 ; SERVICE_KERNEL_DRIVER
StartType = 3 ; SERVICE_DEMAND_START
ErrorControl = 1 ; SERVICE_ERROR_NORMAL
ServiceBinary = %12%\usbser.sys
LoadOrderGroup = Base
[Strings]
CCS = "FI, UNAM"
CCS_CDC = "Electronic Systems USB-Serial Interface"
Serial.SvcDesc = "USB to RS-232 driver"
```

Cuando se posee este controlador dentro de la PC, la conexión entre el microcontrolador y la PC se configurará de manera automática, asignándose por default un *puerto COM* de la PC para la gestión de la comunicación. Además todo ello se llevará a cabo desde el momento en

que el microcontrolador sea conectado al puerto *USB* del equipo de cómputo. Para conocer el puerto COM asignado al microcontrolador solo es necesario acceder al Administrador de Dispositivos desde el Panel de Control de Windows y ver las propiedades del microcontrolador de manera similar a la descrita previamente para el teléfono celular. La figura 5.12 representa una imagen muestra de dicho procedimiento.

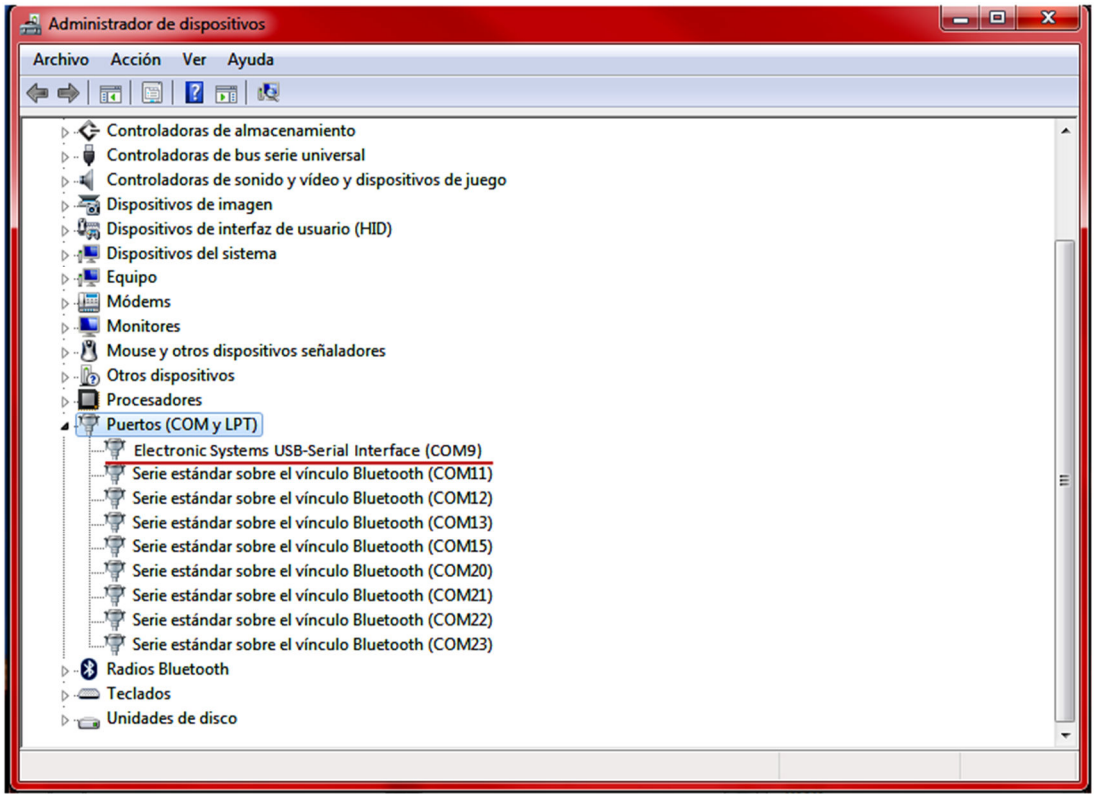


Figura 5.12. Ventana del administrador de dispositivos de Microsoft Windows 7.

Por lo tanto de manera similar a la comunicación con el teléfono móvil, los comandos que son implementados para la transmisión y recepción de datos provenientes del microcontrolador son los siguientes:

```
MSComm1.CommPort = 9                 'Se especifica el valor del puerto COM para la conexión con el teléfono.
MSComm1.Settings = "115200,N,8,1"    'Se configuran las características de comunicación con el mismo.
MSComm1.InputMode = InputModeText    'Los datos se transmitirán como texto en código ASCII.
MSComm1.PortOpen = True              'Se abre el puerto indicado y se establece la comunicación con el dispositivo.
MSComm1.Output = "D"                 'Se envía el carácter ´D´ hacia el microcontrolador.
var = MSComm1.Output                 'Se guarda en la variable ´var´ el dato proveniente del microcontrolador.
```

Y de igual manera con base en estos comandos se lleva a cabo todo el intercambio de información entre el microcontrolador y la PC.

5.4.4. RECONOCIMIENTO DE LAS SEÑALES DTMF.

Una vez que la PC recibe los datos obtenidos del proceso de muestreo, que a su vez son recopilados y enviados por el microcontrolador, ésta realiza una DFT de los mismos y al mismo tiempo almacena los datos resultantes en un arreglo. Cuando se copiaron los resultados de este arreglo y se graficaron con ayuda de MATLAB, estos arrojaron la posición precisa, dentro del arreglo, que ocupan las espigas correspondientes a las dos frecuencias que componen las señales DTMF.

De tal forma que la primera espiga puede ocupar solamente las posiciones [20, 22, 24, 27] del arreglo, correspondientes a las frecuencias [697, 770, 852, 941] Hertz respectivamente. Mientras que la segunda espiga se puede encontrar únicamente en las posiciones [34, 38, 42] correspondientes a las frecuencias [1209, 1336, 1477] Hertz. La figura 5.13 ejemplifica este caso.

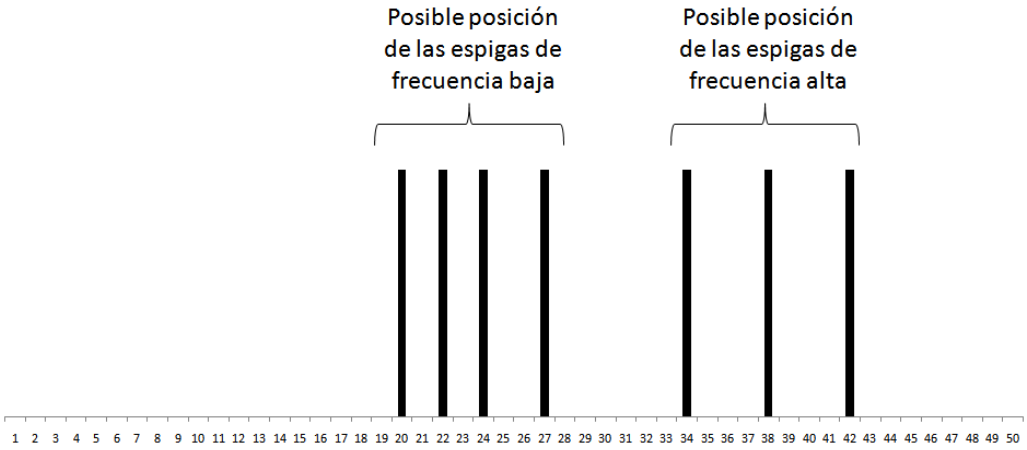


Figura 5.13. Gráfica de los posibles lugares de las espigas de frecuencia conforme al arreglo de datos obtenido.

Registrando los resultados en la tabla 5.1 se puede obtener fácilmente la combinación de posiciones en el arreglo, las cuales determinan la señal DTMF que representan. Resulta entonces muy útil compararla con la figura 4.12 del capítulo anterior.

K	34	38	42
20	1	2	3
22	4	5	6
24	7	8	9
27	*	0	#

Tabla 5.1. Relación entre las posiciones en el arreglo de datos y las teclas que representan su combinación.

Una vez que se conocen estos datos con anticipación, resulta sencillo y útil implementar un algoritmo para encontrar los máximos valores en un arreglo, además de acotar el mismo para reducir tiempo de procesamiento, y de esta manera encontrar la posición exacta de las

dos espigas que representan las frecuencias la señal. La figura 5.14 muestra el algoritmo programado para realizar dicha labor.

En tanto, el código necesario para ejecutar dicho algoritmo y encontrar la primera espiga se muestra a continuación:

```
Frec1 = 19                      'Se inicializa la variable 'Frec1' con el valor de 19.
For k = 20 To 30                'Se programa e inicializa un ciclo FOR que incrementa la variable 'k' desde 20 hasta 30.
If Espec(k) > Espec(Frec1) Then 'Si el valor del dato almacenado en la posición 'k' del arreglo es mayor al
Frec1 = k                       'almacenado con anterioridad, entonces realiza el remplazo para almacenar
End If                          'el dato mayor del arreglo en esta rango de posiciones.
Next k
```

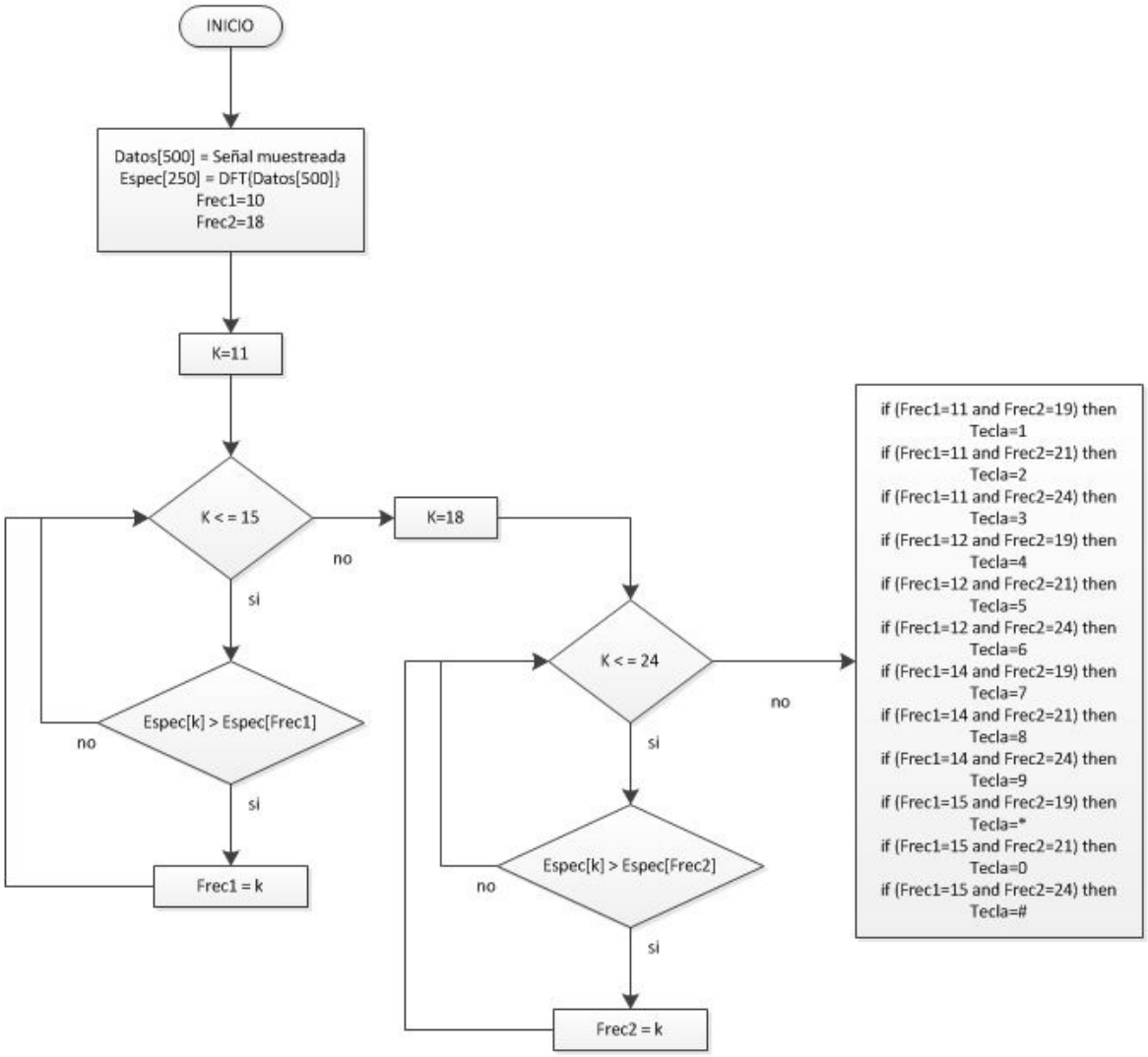


Figura 5.14. Algoritmo de decodificación de señales DTMF.

Para la segunda espiga adecuamos los valores de búsqueda, de tal forma que el código queda como se muestra a continuación:

```
Frec2 = 33                          'Se inicializa la variable 'Frec2' con el valor de 33.
For k = 34 To 44                    'Se programa e inicializa un ciclo FOR que incrementa la variable 'k' desde 34 hasta 44.
If Espec(k) > Espec(Frec2) Then     'Si el valor del dato almacenado en la posición 'k' del arreglo es mayor al
Frec2 = k                           'almacenado con anterioridad, entonces realiza el remplazo para almacenar
End If                              'el dato mayor del arreglo en esta rango de posiciones.
Next k
```

La variable *Espec()* es el arreglo en donde se encuentran almacenados los datos resultantes del proceso de DFT. Por lo tanto una vez que se obtienen las posiciones exactas de las espigas, almacenadas en las variables *Frec1* y *Frec2*, solo resta ejecutar en cadena una serie de condiciones IF para el reconocimiento las frecuencias. De tal manera:

```
If Frec1 = 20 And Frec2 = 34 Then
    Tecla = "1"                     'Se detecta la tecla '1'.
End If
If Frec1 = 20 And Frec2 = 38 Then
    Tecla = "2"                     'Se detecta la tecla '2'.
End If
If Frec1 = 20 And Frec2 = 42 Then
    Tecla = "3"                     'Se detecta la tecla '3'.
End If
If Frec1 = 22 And Frec2 = 34 Then
    Tecla = "4"                     'Se detecta la tecla '4'.
End If
If Frec1 = 22 And Frec2 = 38 Then
    Tecla = "5"                     'Se detecta la tecla '5'.
End If
If Frec1 = 22 And Frec2 = 42 Then
    Tecla = "6"                     'Se detecta la tecla '6'.
End If
If Frec1 = 24 And Frec2 = 34 Then
    Tecla = "7"                     'Se detecta la tecla '7'.
End If
If Frec1 = 24 And Frec2 = 38 Then
    Tecla = "8"                     'Se detecta la tecla '8'.
End If
If Frec1 = 24 And Frec2 = 42 Then
    Tecla = "9"                     'Se detecta la tecla '9'.
End If
If Frec1 = 27 And Frec2 = 34 Then
    Tecla = "*"                     'Se detecta la tecla '*'.
```

```
End If
If Frec1 = 27 And Frec2 = 38 Then
    Tecla = "0"                    'Se detecta la tecla '0'.
End If
If Frec1 = 27 And Frec2 = 42 Then
    Tecla = "#"                    'Se detecta la tecla '#'.
End If
```

El código completo tanto de la PC como del microcontrolador para la elaboración del presente proyecto se muestra al final de este texto en el apartado ***Anexos***.

5.5. CONEXIÓN DE LOS PERIFÉRICOS DE SALIDA.

A continuación se procederá a detallar más a fondo las conexiones y los circuitos que componen los periféricos que se implementaron en este proyecto.

El sistema está diseñado para que la activación y desactivación de los dispositivos se lleve a cabo por medio del cambio de estado de un bit de control del microcontrolador, por lo tanto, pueden implementarse circuitos y dispositivos alternos para realizar la misma función. En este proyecto se trataron de diseñar e implementar los dispositivos más sencillos que permitieran la funcionalidad deseada.

5.5.1. CONEXIÓN DE LAS LUCES Y LA ALARMA SONORA.

La figura 5.15 muestra la conexión de los dispositivos implementados para la labor de las luces y la alarma sonora del sistema de seguridad. El principal objetivo de este módulo es alertar de forma estridente a las personas dentro y fuera de la propiedad, sobre alguna intromisión dentro de la misma.

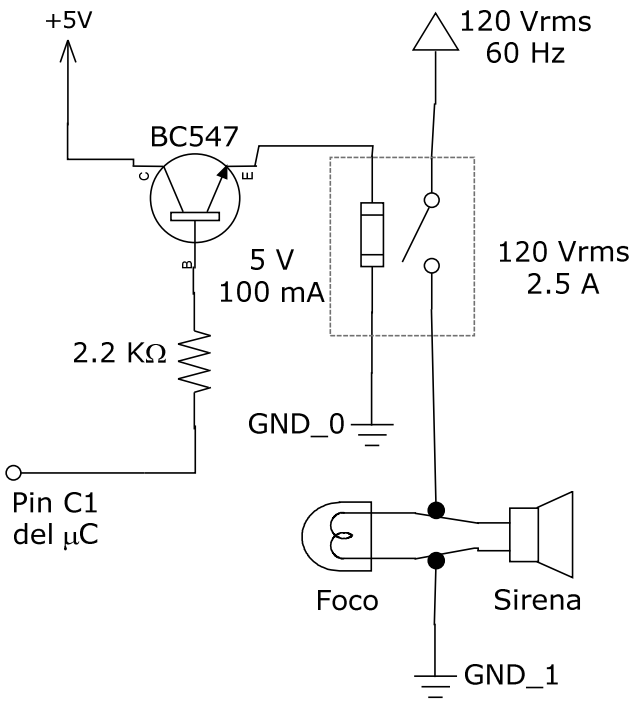


Figura 5.15. Módulo de alerta visual y sonora del sistema de seguridad.

5.5.2. CONEXIÓN DEL INTERCOMUNICADOR O ALTAVOZ.

Para la conexión del intercomunicador del sistema de seguridad se empleará el circuito integrado TDA2822. Este chip integra un amplificador dual que permite el manejo de dos señales eléctricas de audio-frecuencia al mismo tiempo, dadas estas cualidades se puede adaptar dicho dispositivo para fungir como un intercomunicador dentro del sistema de seguridad. La figura 5.16 muestra el diagrama de conexiones implementado para dicho objetivo.

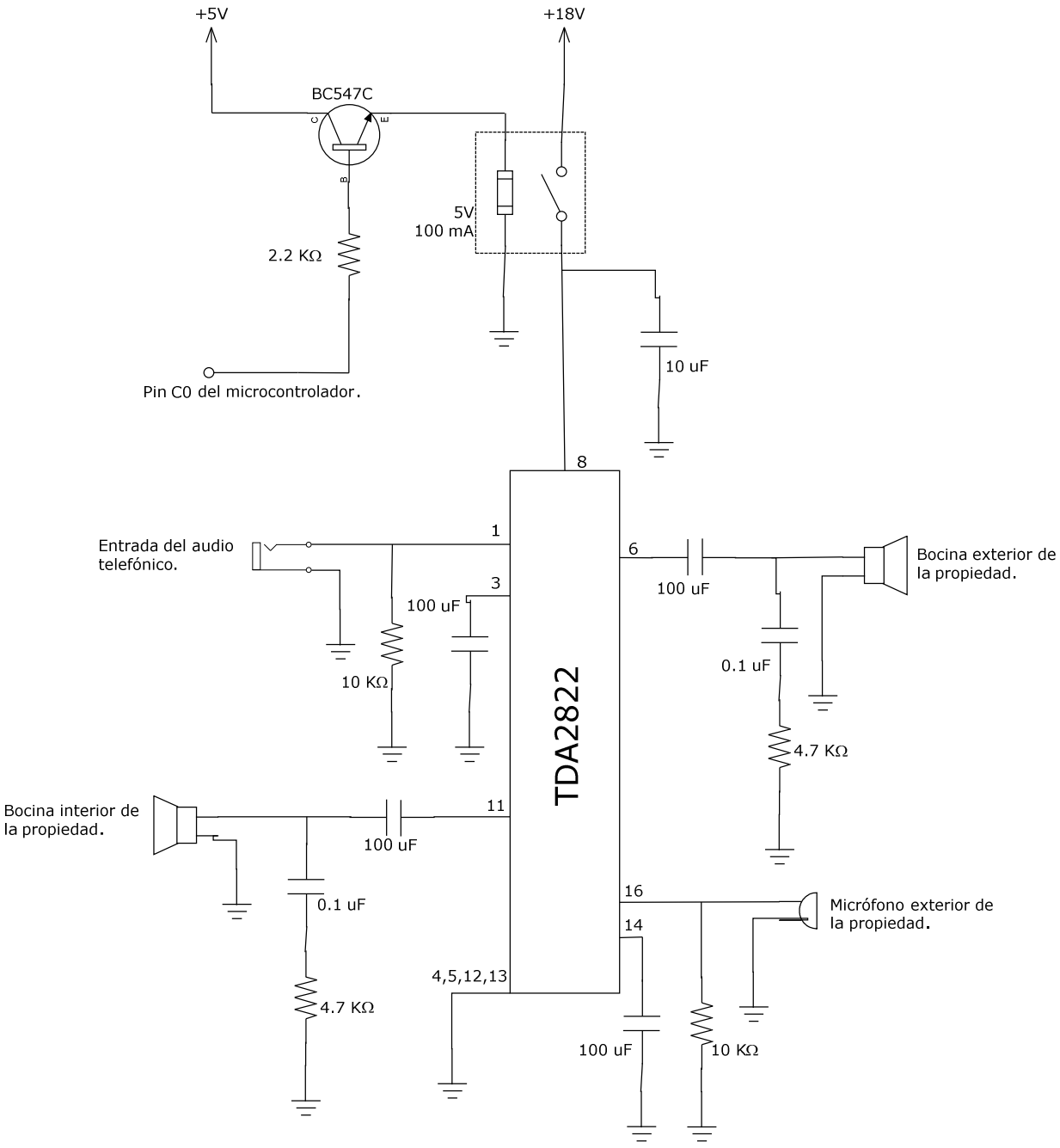


Figura 5.16. Conexión del intercomunicador del sistema.

5.5.3. CONEXIÓN DE LA CHAPA ELECTRÓNICA.

La figura 5.17 muestra el circuito de control de apertura de la puerta principal de la propiedad.

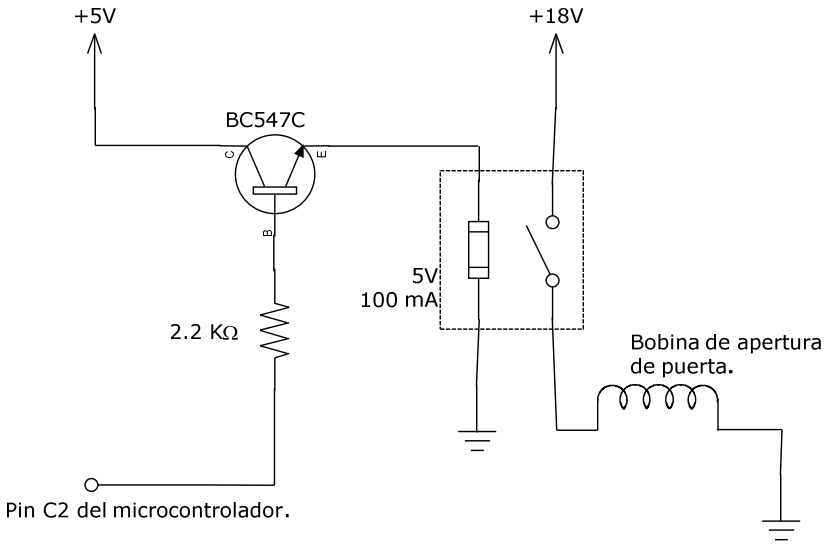


Figura 5.17. Conexión del circuito para la chapa electrónica.

5.6. RESUMEN Y SÍNTESIS.

El presente capítulo se enfocó en los aspectos prácticos del mismo, tales como la configuración e instalación de todos los componentes y dispositivos que conforman el sistema.

Se detallaron más a fondo los circuitos, las conexiones, los archivos y procesos de configuración de los dispositivos, el código de programación para el control de la comunicación entre la PC y el microcontrolador, el código para el reconocimiento de las señales DTMF y los algoritmos diseñados para dicho proceso.

De esta manera queda completa la descripción del diseño e implementación del sistema de seguridad y únicamente resta detallar los resultados experimentales obtenidos para todos y cada uno de los procesos involucrados en el desarrollo del sistema, y los resultados obtenidos durante la implementación del mismo, lo cual es descrito en el siguiente capítulo.

CAPÍTULO 6.
PRUEBAS Y RESULTADOS.

El objetivo del presente capítulo es detallar la experimentación realizada a todos los procesos involucrados en el sistema de seguridad desarrollado en este proyecto de tesis. Se tratará de describir de manera lógica dicho proyecto, es decir, comenzando desde la parte de detección hasta finalizar con los periféricos de salida del sistema.

Algunos procesos de experimentación resultaron más sencillos que otros, y existen algunos con mayor relevancia, esto dado que ciertas mediciones se llevaron a cabo con un sentido meramente práctico. Tal vez el mejor ejemplo de un proceso de experimentación de gran relevancia es el que se llevó a cabo para el procesamiento de las señales DTMF, que generó el parámetro definido sobre tiempo de muestreo que se implementó para el sistema de seguridad.

6.1. PROCESAMIENTO DE LAS SEÑALES DTMF.

Las pruebas que a continuación se detallan fueron implementadas para encontrar una frecuencia de muestreo óptima que facilitara el reconocimiento de las señales DTMF del audio telefónico.

Es importante mencionar que ciertos parámetros de procesamiento tuvieron que tomar un valor fijo de manera obligada, el caso más importante de dicha situación se presentó con el número de muestras que el microcontrolador PIC18F4550 es capaz de recopilar. Puesto que el guardar datos en la memoria Flash (EEPROM) es un proceso que conlleva varios milisegundos (aproximadamente de 5 a 10 ms por dato) para el microcontrolador utilizado, por lo que la velocidad para la adquisición de datos se encuentra muy restringida, provocando una frecuencia de muestreo muy baja, por lo que no resta más que utilizar la memoria de programa (RAM) para dicha labor.

Para el procesamiento del audio se decidió tomar como parámetro de muestreo un valor de N=500 muestras, cantidad que se considera factible para el objetivo requerido, además de que no satura la memoria de programa dando de esta manera un espacio para el aumento de código de control en caso de que fuese necesario. La figura 6.1 muestra la ventana de compilación indicando el estado de las memorias RAM y ROM.

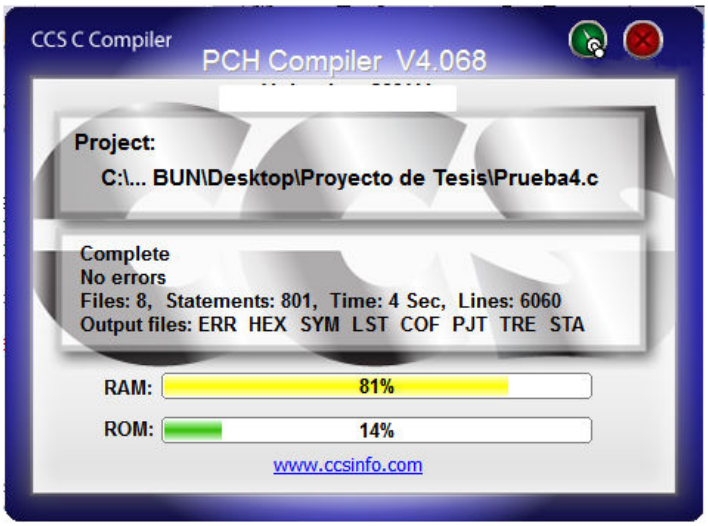


Figura 6.1. Ventana de compilación del CCS PIC C Compiler.

Como se analiza a continuación el número de muestras N afecta directamente a la resolución de la DFT, y por ende al procesamiento del audio telefónico del sistema.

La resolución de una señal discreta (Δf_{DFT}) se encuentra definida por la relación entre la frecuencia de muestreo f_s y el número de muestras *N* de la señal, tal como se muestra en la ecuación 6.1.

$$\Delta f_{DFT} = \frac{f_s}{N} \tag{6.1}$$

Por lo que si una señal se compone, por ejemplo de dos senoidales simples, y tiene sus componentes en frecuencia relativamente cerca una de la otra, tal como se muestra en la figura 6.2, será más complejo detectar independientemente una u otra debido a la pequeña separación que existe entre las espigas que representan dichas componentes, este efecto es conocido como "*Aliasing*".

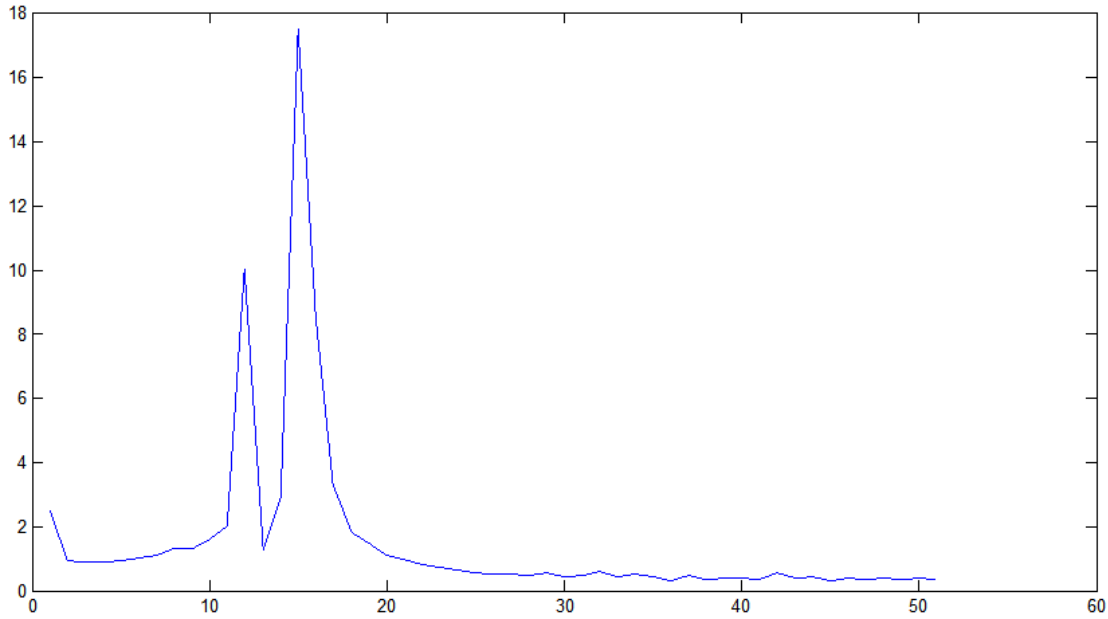


Figura 6.2. Ejemplo de la DFT de una señal DTMF.

Con base en la ecuación 6.1, existen dos formas de incrementar el espacio entre espigas facilitando de esta manera la detección de cada una de ellas. La primera es aumentar el número de muestras *N*, sin embargo, ésta solución no es viable dado que la memoria disponible para la recopilación de datos se encuentra restringida, por lo que *N* se mantendrá constante en un valor de 500. La segunda, y que se aplicó en este proyecto, es reducir la frecuencia de muestreo f_s de manera que la resolución se reduzca y el espacio entre espigas crezca. Por supuesto, este decremento en la frecuencia de muestreo debe respetar el teorema de *Nyquist* para evitar el *aliasing* y la pérdida de información.

Al modificar la frecuencia de muestreo no sólo se incrementa o decrementa el espacio entre espigas, sino que también las desplaza dentro del eje horizontal representado por la variable *N* (número de muestras). Véase la figura 6.3.

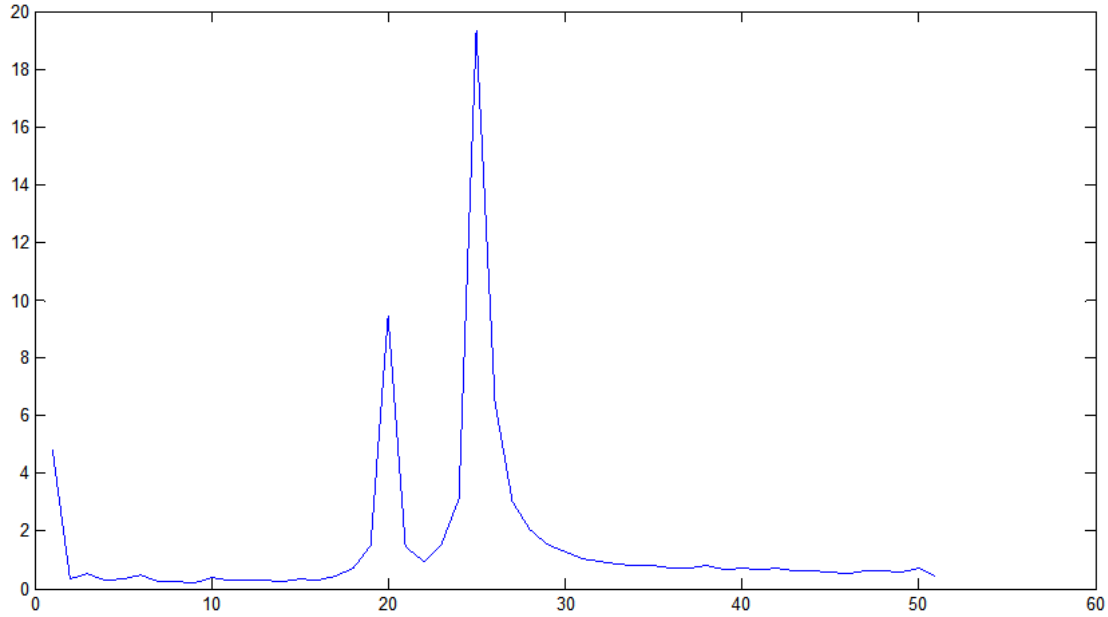


Figura 6.3. Misma señal DTMF muestreada a una frecuencia menor.

Existen dos consecuencias muy claras que afectan directamente el diseño del sistema para la detección de las señales DTMF. La primera, que es conveniente muestrear a una frecuencia lo suficientemente baja para detectar claramente las espigas de frecuencia de la señal, pero tampoco tan baja como para que exista pérdida de información. Otro aspecto igual de importante y muy sencillo de deducir, es que una frecuencia de muestreo muy baja, impactará directamente en el tiempo que el sistema tarda en realizar el proceso de detección de las componentes en frecuencia de la señal en cuestión.

6.1.1. SELECCIÓN DE LA FRECUENCIA DE MUESTREO.

La resolución se define como el cambio o incremento más pequeño que es capaz de detectar cierto dispositivo. Para el caso en cuestión, la resolución de la DFT es el intervalo mínimo de frecuencia que presentan los puntos que conforman dichas gráficas. En algunos otros campos, como medición e instrumentación, el concepto de resolución también es conocido como sensibilidad del dispositivo o instrumento.

Es necesario tomar en cuenta de igual manera que el incremento mínimo en frecuencia que existe entre dos distintas señales DTMF es igual a 73 Hz (véase la figura 4.12), entonces la

resolución de la DFT debe ser menor a este intervalo de frecuencias, dado que de otra forma existiría el riesgo de no poder diferenciar una componente de la otra durante el proceso de detección.

Para una frecuencia de muestreo de 50 KHz:

$$\Delta f_{DFT} = \frac{50[KHz]}{500} = 100\ Hz \tag{6.2}$$

Esto nos indica que un incremento unitario en el eje k equivale a un incremento de 100 Hz en el espectro discreto (Resolución de 100 Hz). Por lo que teóricamente una frecuencia de muestreo de 50 KHz podría provocar errores durante el proceso de detección.

Ahora para una frecuencia de muestreo de 25 KHz:

$$\Delta f_{DFT} = \frac{25[KHz]}{500} = 50\ Hz \tag{6.3}$$

Entonces una frecuencia de muestreo menor o igual a 25 KHz sí produciría una correcta resolución de DFT para ser implementada durante el proceso de detección de las señales DTMF.

Ahora para la frecuencia de muestreo *fs* de 16.7 KHz:

$$\Delta f_{DFT} = \frac{16.7[KHz]}{500} = 33.4\ Hz \tag{6.4}$$

Por lo que en teoría, la separación mínima entre espigas adyacentes debe ser alrededor de 2 o 3 unidades en el eje *k*. Y aunque en el eje *k* las espigas se separen o se desplacen (Compárense nuevamente las gráficas 6.2 y 6.3), no significa que las frecuencias se modifiquen, es muy importante aclarar que lo único que está cambiando es la resolución de la DFT y por ende la gráfica resultante. En ambas, la espiga de frecuencia menor representa a la componente de 941 Hz y la de frecuencia mayor a la componente de 1209 Hz (*Tecla '*'*).

A continuación se mostrarán gráficas de todas las señales DTMF, muestreadas a diferentes frecuencias, pero antes se detallará el significado de los puntos y valores especificados en dichas gráficas.

F_{HR}: Valor real de la frecuencia correspondiente a la espiga superior.
F_{HO}: Valor obtenido en la gráfica que corresponde a la espiga superior.
$\Delta e_H = |F_{HR} - F_{HO}|$: Error entre el valor real y el valor obtenido.

F_{LR}: Valor real de la frecuencia correspondiente a la espiga inferior.
F_{LO}: Valor obtenido en la gráfica que corresponde a la espiga inferior.
$\Delta e_L = |F_{LR} - F_{LO}|$: Error entre el valor real y el valor obtenido.

6.1.1.1. DÍGITO 1 DEL TECLADO TELEFÓNICO.

La señal DTMF correspondiente al dígito 1 posee sus componentes en 697 y 1209 Hz.

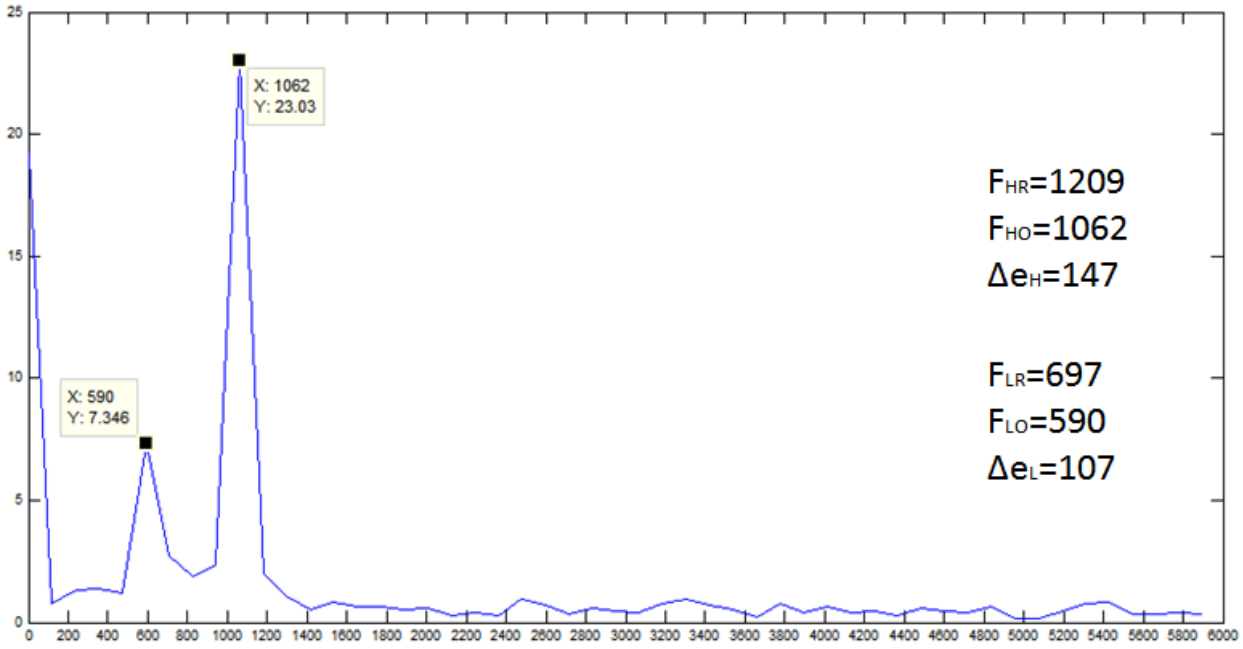


Figura 6.4. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 59[KHz].

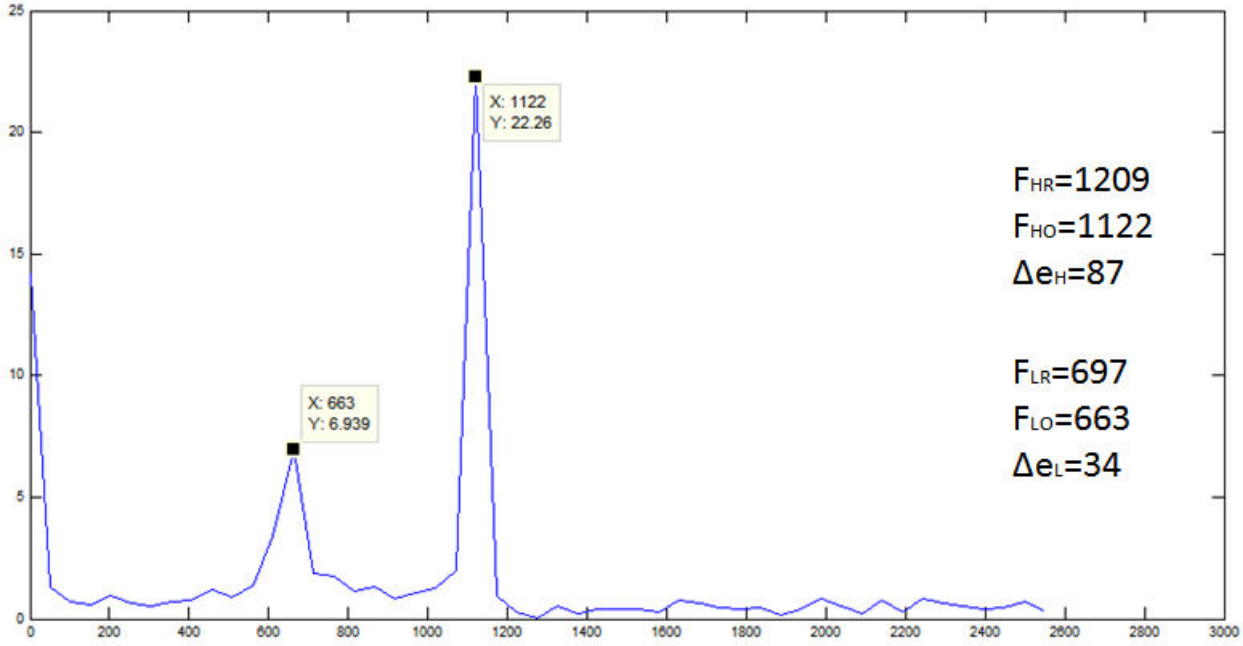


Figura 6.5. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 25.5[KHz].

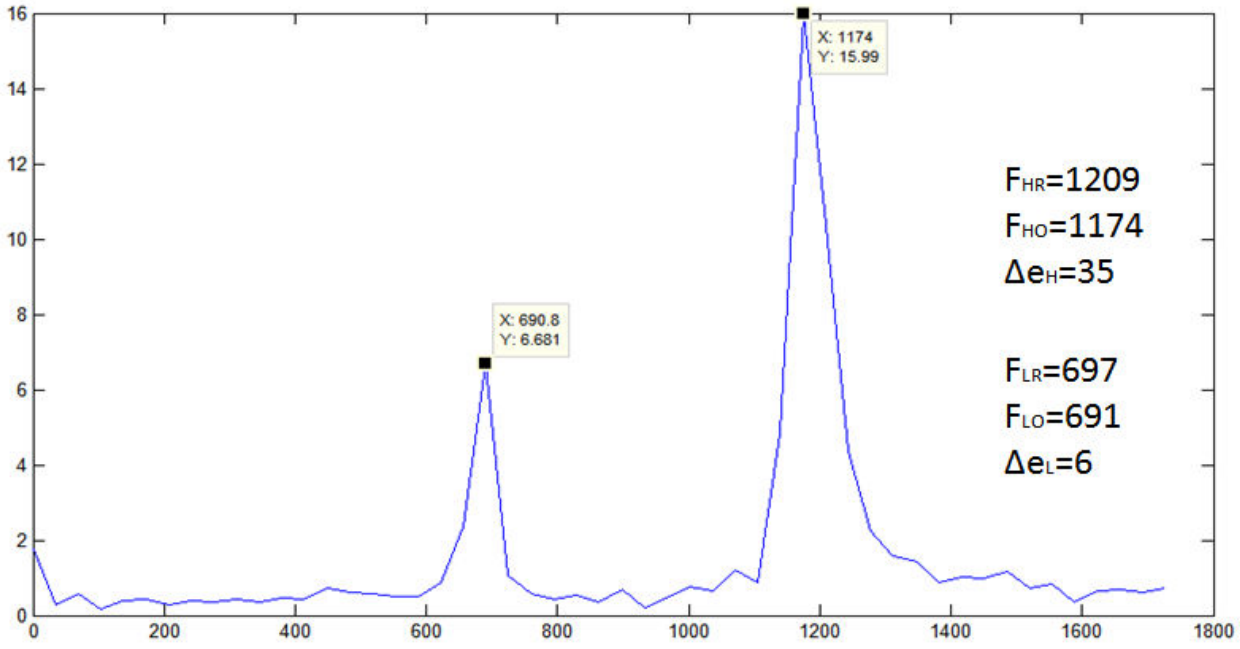


Figura 6.6. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 16.7[KHz].

6.1.1.2. DÍGITO 5 DEL TECLADO TELEFÓNICO.

La señal DTMF correspondiente al dígito 5 posee sus componentes en 770 y 1336 Hz.

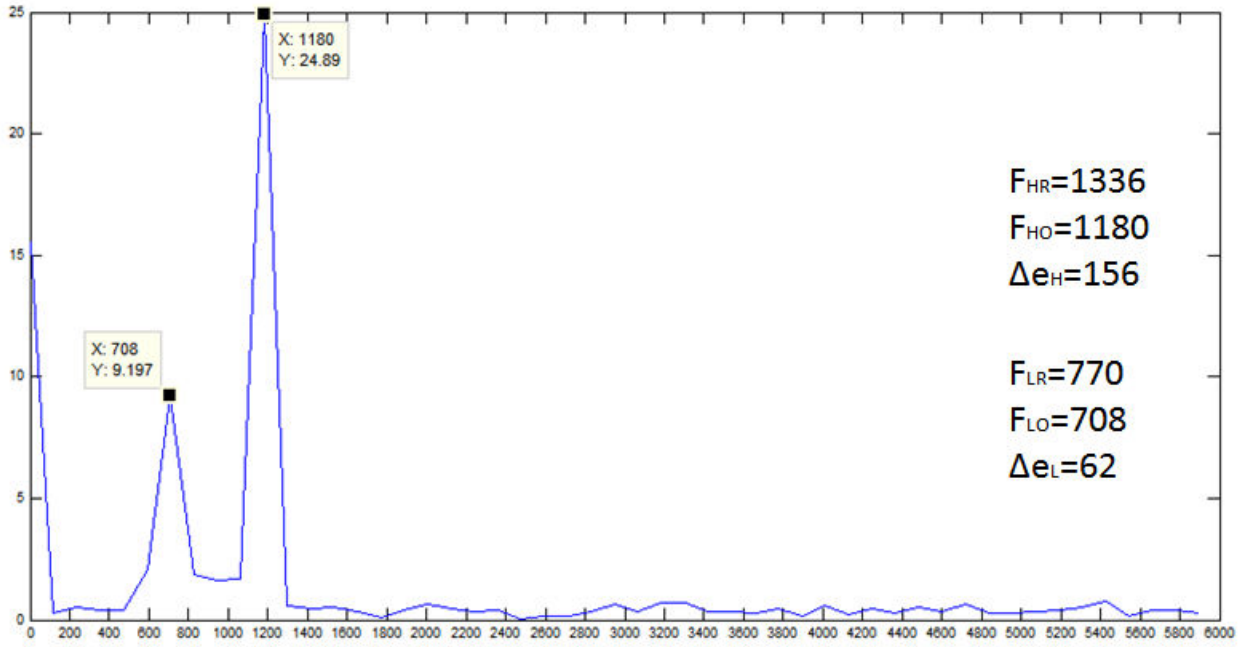


Figura 6.7. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 59[KHz].

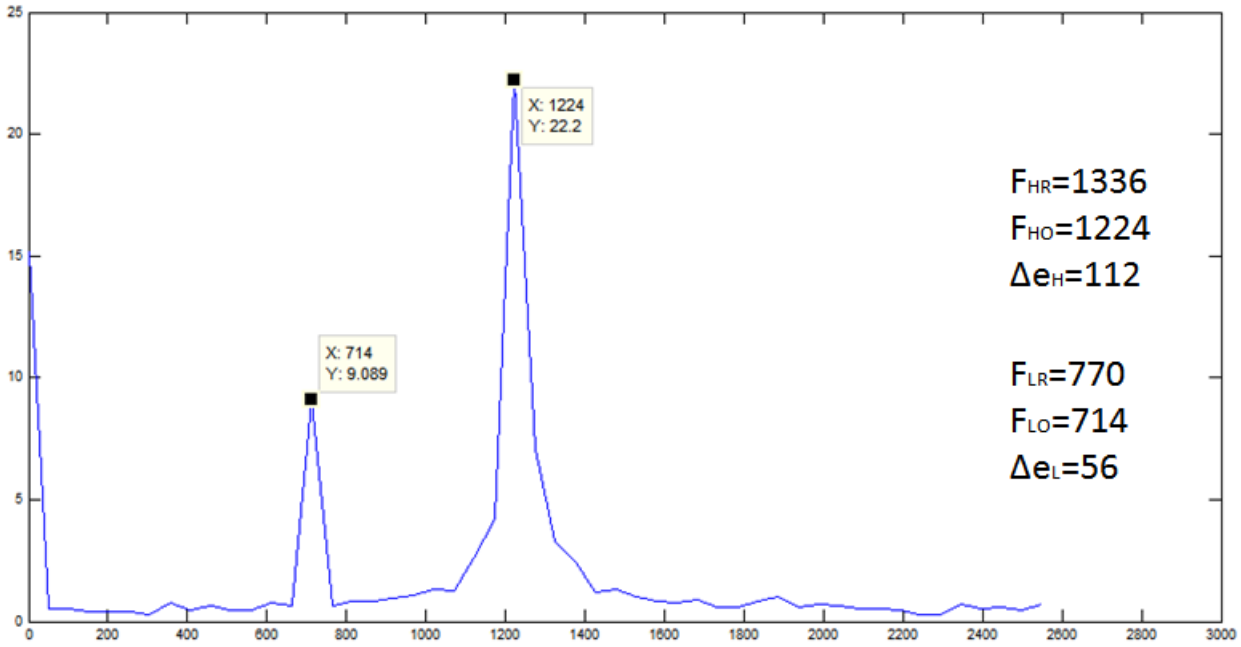


Figura 6.8. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 25.5[KHz].

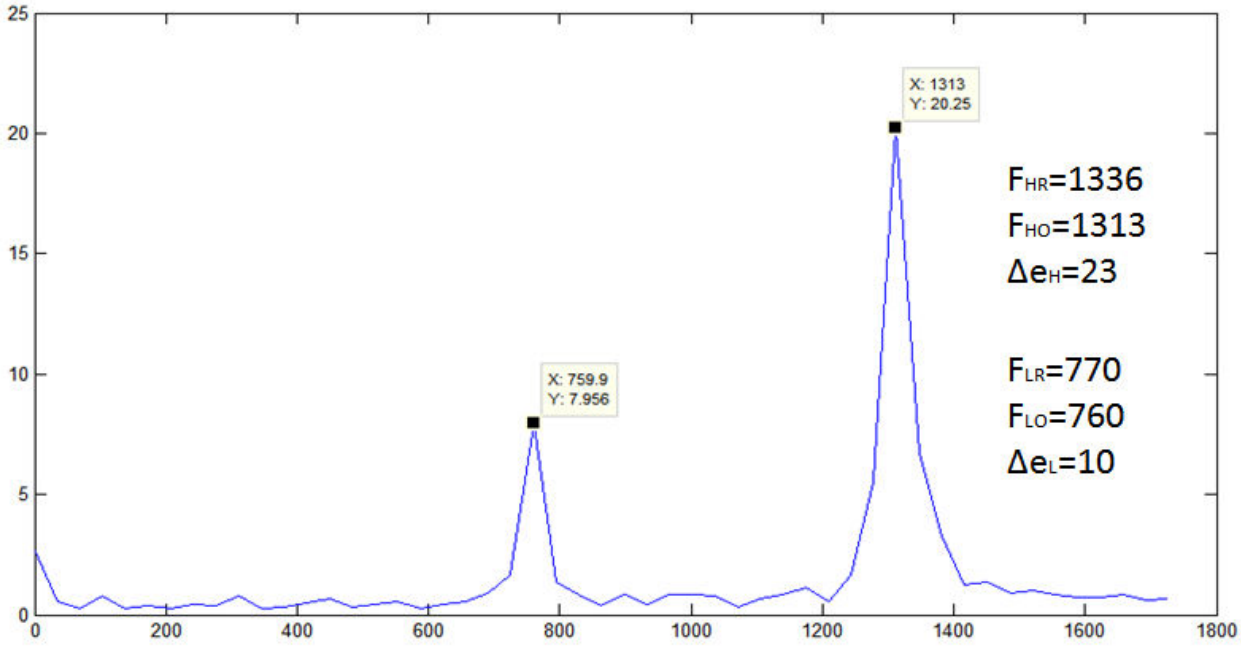


Figura 6.9. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 16.7[KHz].

6.1.1.3. DÍGITO 9 DEL TECLADO TELEFÓNICO.

La señal DTMF correspondiente al dígito 9 posee sus componentes en 852 y 1477 Hz.

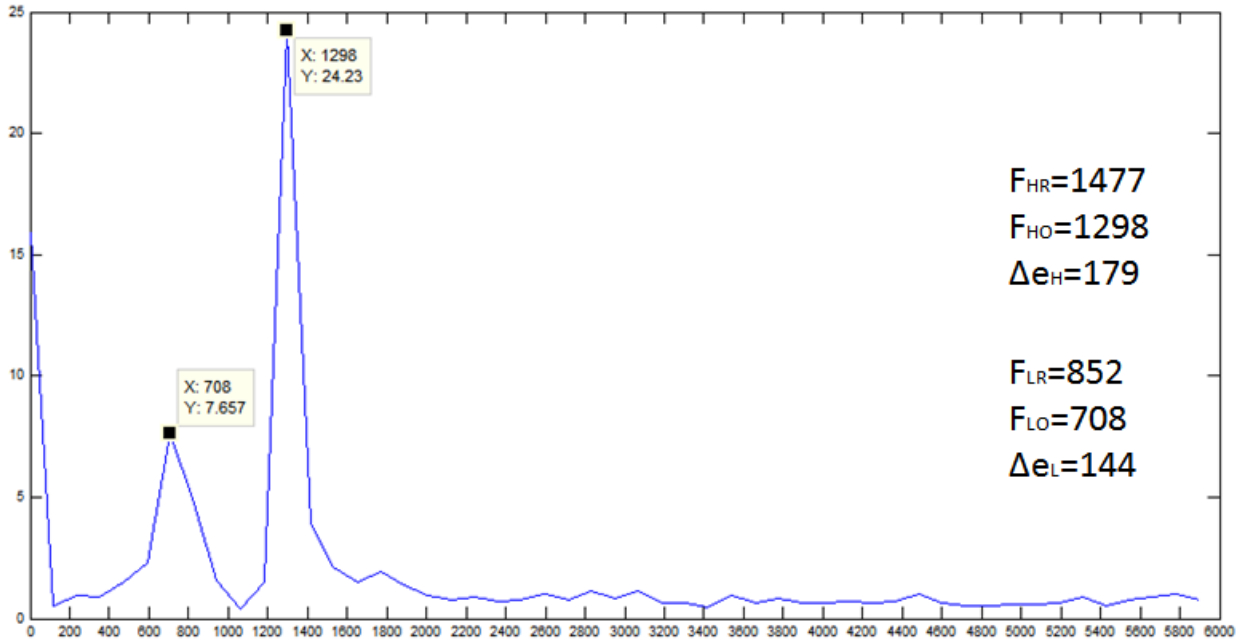


Figura 6.7. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 59[KHz].

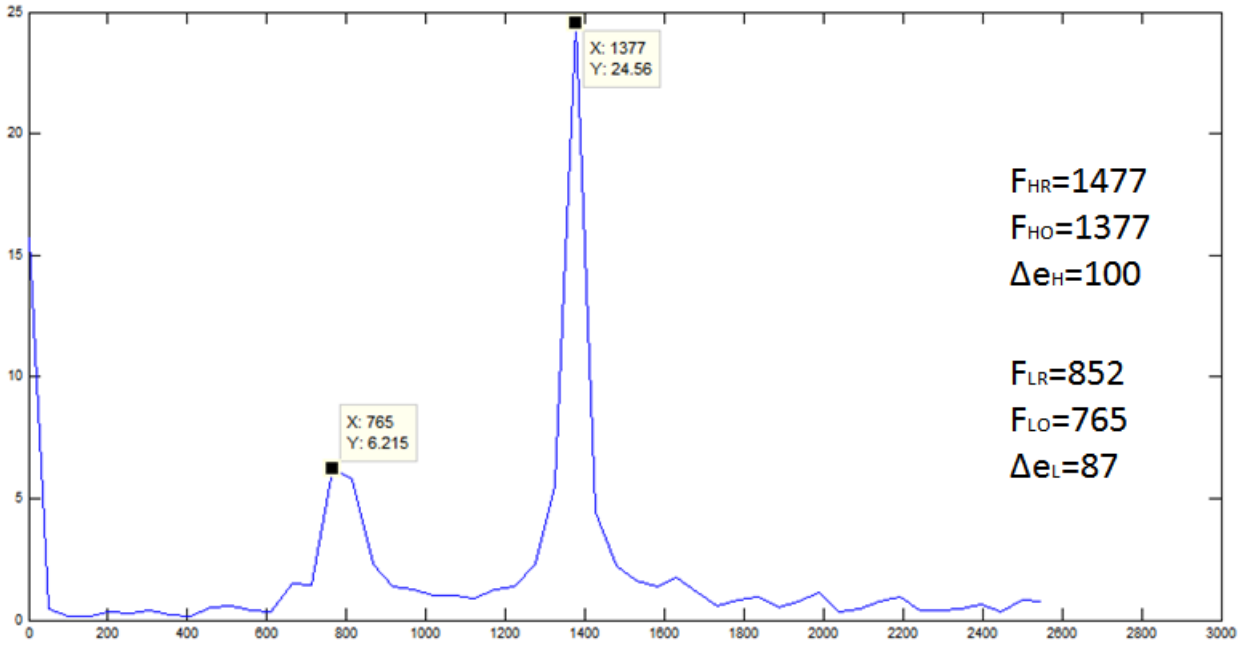


Figura 6.8. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 25.5[KHz].

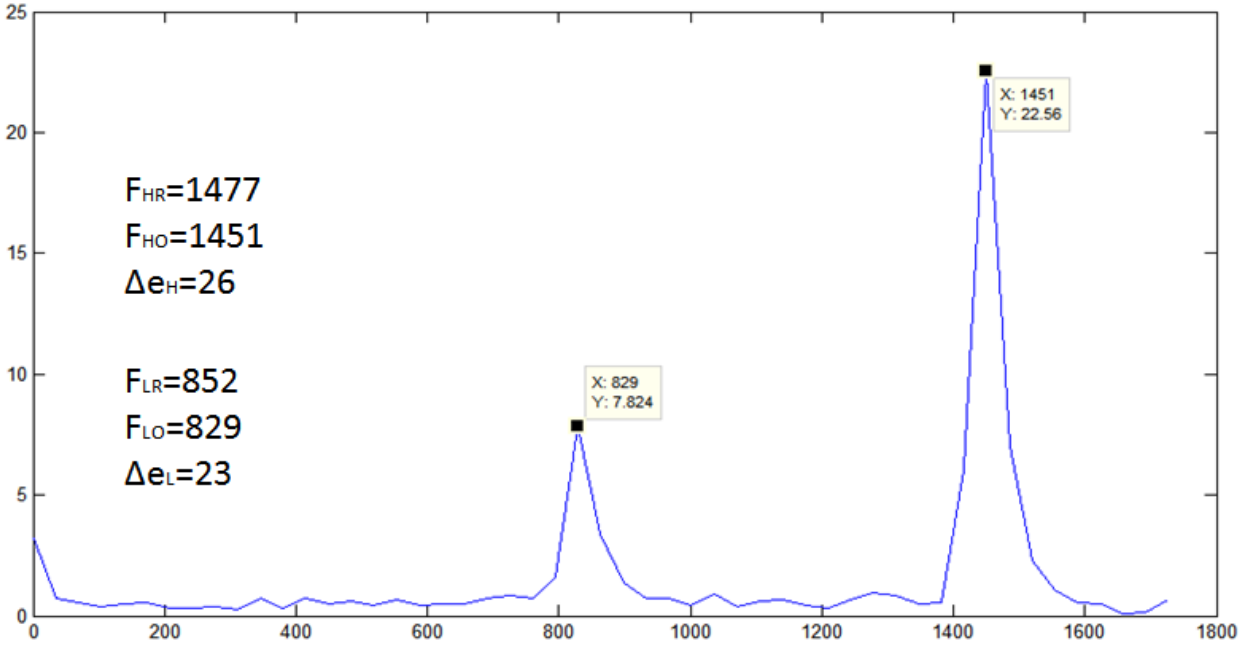


Figura 6.9. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 16.7[KHz].

6.1.1.4. DÍGITO 0 DEL TECLADO TELEFÓNICO.

La señal DTMF correspondiente al dígito 0 posee sus componentes en 941 y 1336 Hz.

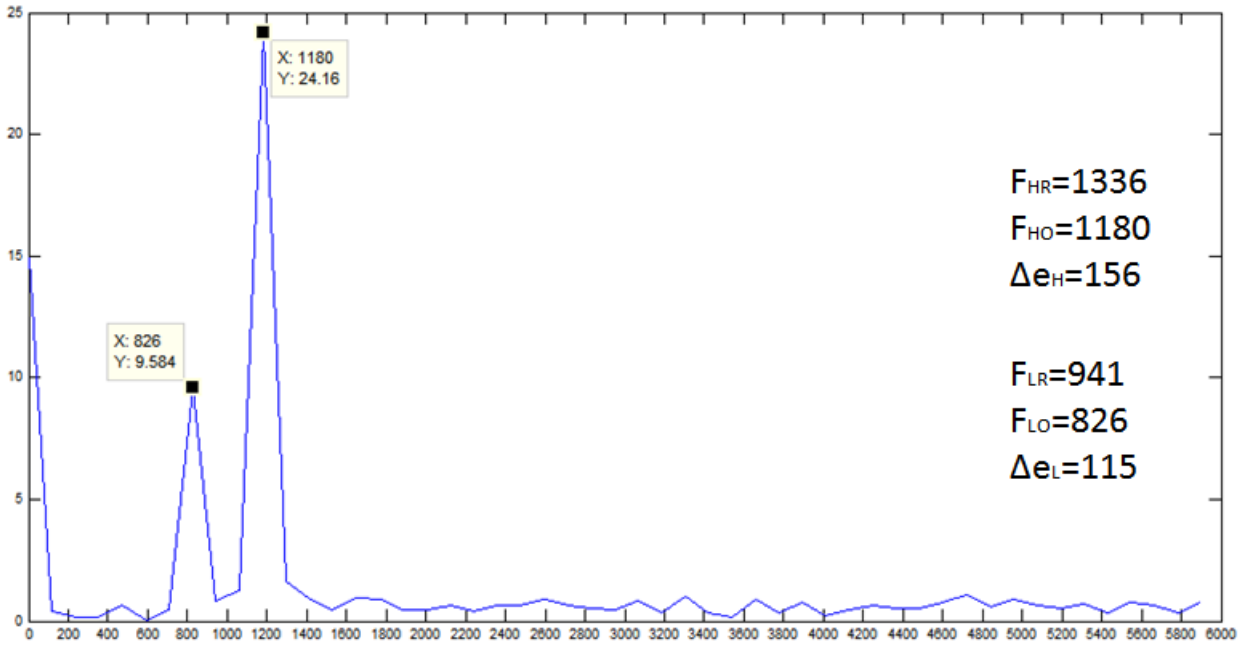


Figura 6.7. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 59[KHz].

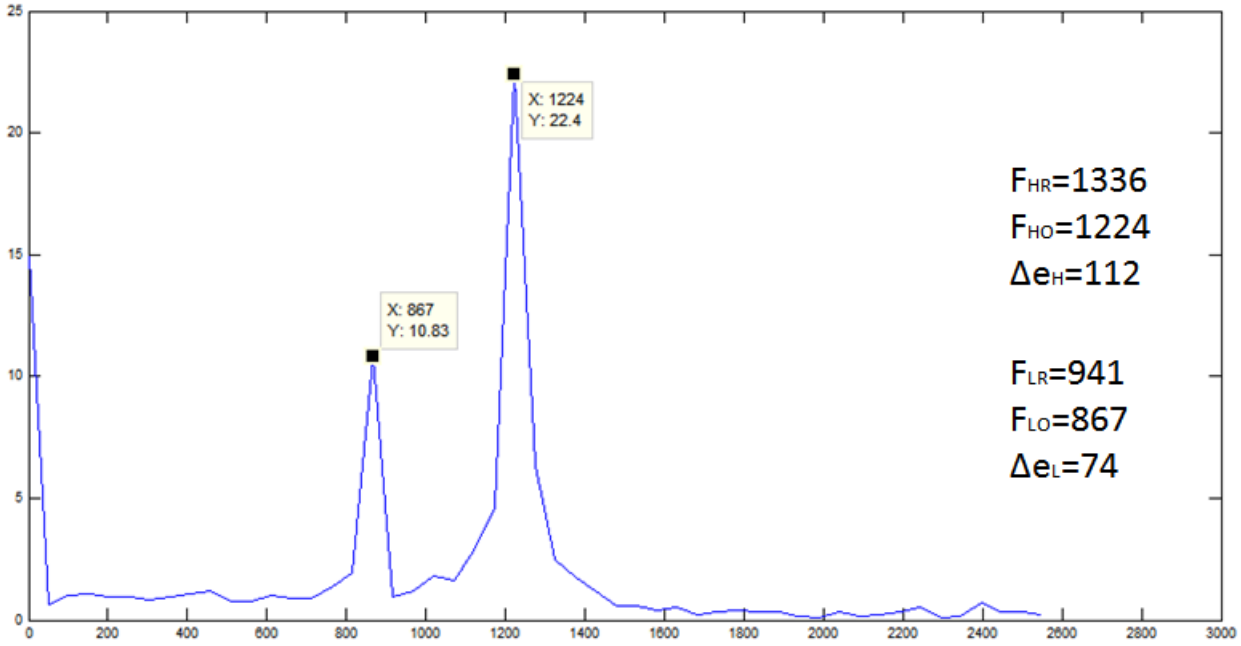


Figura 6.8. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 25.5[KHz].

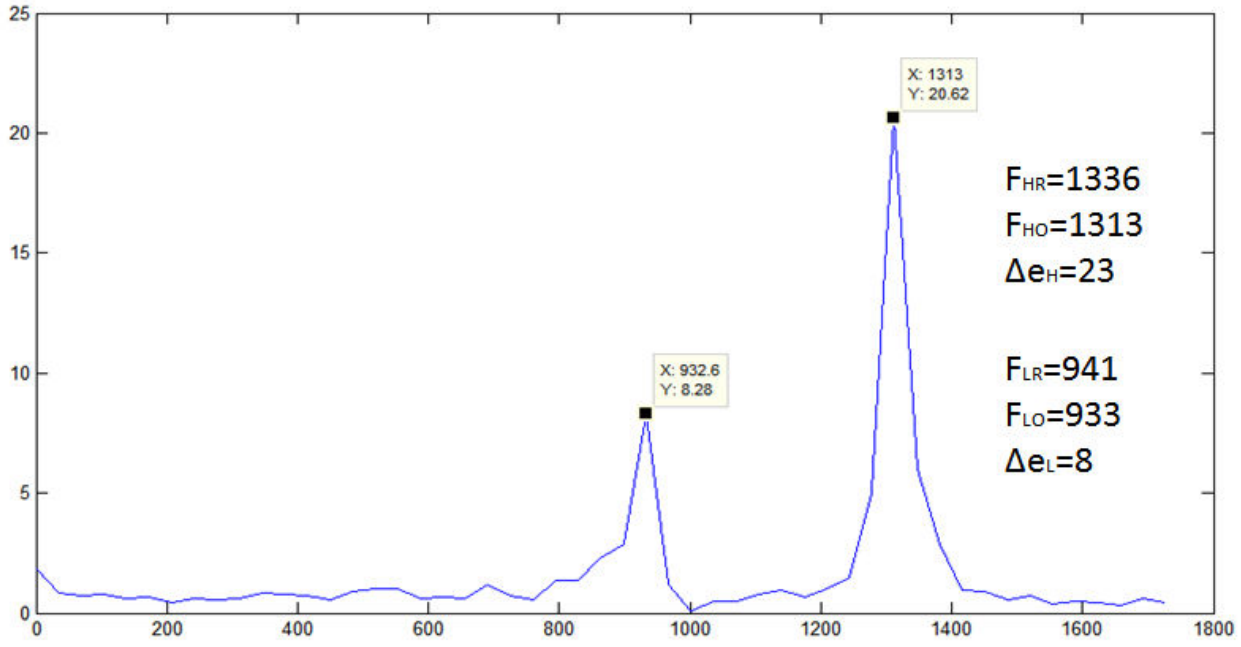


Figura 6.9. Gráficas de la señal en el dominio de la frecuencia, con una tasa de muestreo de 16.7[KHz].

De las figuras mostradas anteriormente, nótense las distancias que separan las espigas que representan las componentes en frecuencia de las señales, como varían en su posición y distanciamiento dichas espigas conforme reducimos la frecuencia de muestreo, conservando el número total de muestras. Por lo tanto considérese la complejidad que puede conllevar el encontrar mediante un algoritmo lógico la posición exacta de las espigas, para el reconocimiento de las señales DTMF, cuando éstas no se encuentran a una distancia razonable una de la otra.

De la figura 4.13 (Capítulo 4) las señales DTMF se componen de frecuencias situadas en dos diferentes gamas. Una de ellas, la componente de frecuencia baja, puede situarse en 697, 770, 852 ó 941 [Hz]. La componente de frecuencia alta puede tener un valor de 1209, 1336, 1477 ó 1633 [Hz]. Como ya se ha mencionado, tener una resolución de DFT muy grande, es decir una frecuencia de muestreo alta, limitada a un número total de muestras relativamente bajo, como es el caso del sistema en cuestión (N=500), produce que las espigas de frecuencia obtenidas por medio de la DFT tengan una proximidad bastante perjudicial para el objetivo del proceso de detección. Inclusive puede darse el caso de que si las espigas consecuentes no difieren lo suficiente en frecuencia, después de la DFT, en la gráfica resultante no se obtendrá una diferencia en el posicionamiento de la espiga que represente dicho valor.

Puede observarse también que el error que entregan las gráficas previas, con respecto a los valores de frecuencia que representan las componentes de las señales DTMF obtenidas mediante los procesos realizados por el sistema de seguridad, se reduce conforme se disminuye la frecuencia de muestreo. El promedio de errores se muestra a continuación:

$$Para\ \ f_s = 59\ KHz\ \Rightarrow\ \Delta e_H = \frac{(156 + 147 + 179 + 156)[Hz]}{4} = 159.5\ Hz;\quad \Delta e_L = 107\ Hz$$

$$Para\ \ f_s = 25.5\ KHz\ \Rightarrow\ \Delta e_H = 102.75\ Hz;\quad \Delta e_L = 62.75\ Hz$$

$$Para\ \ f_s = 16.7\ KHz\ \Rightarrow\ \Delta e_H = 26.75\ Hz;\quad \Delta e_L = 11.75\ Hz$$

Estos promedios fueron obtenidos con el propósito de identificar más claramente como el error entre el valor real y obtenido de las espigas que componen las señales DTMF, persiguen una tendencia degenerativa conforme se disminuye la frecuencia de muestreo y se mantiene constante el número de muestras.

Lo descrito anteriormente representa la razón principal por la cual se eligió una frecuencia de muestreo de 16.7 KHz.

6.2. PRUEBAS GENERALES DE FUNCIONAMIENTO DEL SISTEMA.

A continuación se presentará la descripción de las diversas situaciones que se pueden presentar como eventos de interés para el sistema de seguridad. Como se detalló anteriormente, cada uno de estos sucesos llevan al sistema a situarse en un estado diferente de control sobre el cual procede a ejecutar una serie de comandos y acciones, los cuales corresponden al tipo de respuesta programada para dicho estado.

Las imágenes que se anexan en los resultados obtenidos pueden no mostrar detalles muy específicos del evento registrado, ya que la intención de las mismas es proveer una sencilla prueba de que el sistema pudo detectar o responder de manera favorable a los diversos sucesos para los que fue programado el sistema de seguridad.

6.2.1. ESTADO DE MONITOREO DEL SISTEMA.

El estado "0" representa al estado de monitoreo del sistema, no es solamente el estado inicial del sistema dado que vuelve a dicho estado en cuanto termina de ejecutar las acciones diseñadas para el resto de estados y sus eventos. Mientras el sistema se encuentre en este estado la única acción que se encontrará realizando es el cambio en el valor de los sensores conectados a uno de los puertos digitales y la posible presencia de señales DTMF en el audio telefónico conectado al convertidor A/D del microcontrolador.

La imagen 6.10 muestra la ventana principal de la aplicación instalada en la PC mientras el sistema se encuentra en el estado de monitoreo.

Figura 6.10. Ventana principal de la aplicación instalada en la PC para el sistema de seguridad.

Como puede observarse el sistema tiene acceso a un *ComboBox* el cual enlista el nombre de varios usuarios que apuntan a sus correspondientes números telefónicos, en este caso el usuario se registró como “Prueba” y se apunta al número del teléfono celular que se usó para realizar las pruebas. A la aplicación se le agregó un fichero que funge como base de datos para el registro de los diferentes usuarios y números de teléfono que se deseen probar para el sistema. Para el propósito de este proyecto no se consideró conveniente detallar más a fondo el funcionamiento de dicho componente debido a ser más un accesorio que un elemento vital en el funcionamiento del sistema de seguridad.

6.2.2. APERTURA NO AUTORIZADA DE ALGÚN ACCESO.

El estado “2” del sistema representa a la apertura no autorizada de uno de los accesos, una puerta o una ventana por ejemplo, se detecta por medio de los sensores ópticos (opto-interruptores) los cuales envían una señal que puede tomar dos niveles de voltaje únicamente 5[V] o GND. Dichos sensores se conectan directamente a uno de los puertos digitales del microcontrolador tal como se indicó en el capítulo anterior de este texto.

Cuando la apertura de uno de estos accesos se realiza, el sensor cambia el nivel de voltaje a su salida, dicho cambio es detectado a su vez por el microcontrolador que a su vez comunica el valor instantáneo del puerto digital a la PC, ésta procesa el dato recibido y con base en el valor del mismo determina qué sensor fue el que se activó, de esta manera la PC cambia de estado de control y procede con la ejecución de los comandos para la cual fue programada hasta que de nueva cuenta regrese a su estado de monitoreo.

La figura 6.11 muestra la ventana emergente del sistema donde se informa que uno de los accesos ha sido abierto sin autorización y se procederá a llamar al usuario para informarle del evento.

Figura 6.11. El sistema envía el comando al teléfono celular para establecer la llamada telefónica al usuario.

La figura 6.12 muestra una imagen de la pantalla del teléfono móvil del sistema realizando la llamada telefónica al usuario, ésta demuestra que los comandos AT enviados desde la PC hacia el teléfono móvil por medio del puerto Bluetooth han sido recibidos y ejecutados favorablemente.

Las dos figuras mencionadas muestran la correcta ejecución programada para el evento que representa la apertura no autorizada de la puerta de la propiedad.

6.2.3. DETECCIÓN DE LA ACTIVACIÓN DEL TIMBRE.

El estado que representa la activación del timbre es el número “1”. El comportamiento de la señal proveniente de ese interruptor es similar al que proviene de los sensores ópticos de la puerta y la ventana. Por esta razón dicha señal es conectada de igual forma al puerto digital del microcontrolador y es procesada en conjunto con las señales de los otros dos sensores.

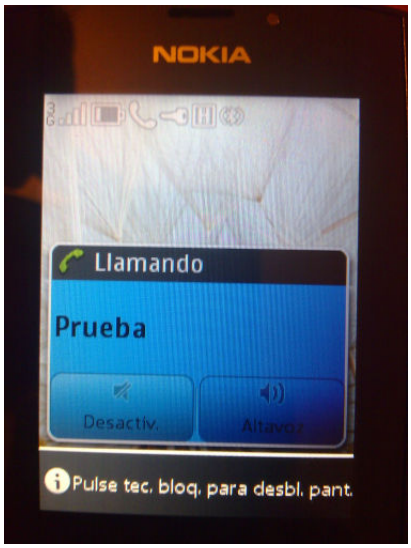


Figura 6.12. Pantalla del teléfono celular realizando la llamada telefónica.

La figura 6.13 muestra las ventanas de la aplicación que aparecen en pantalla cuando el sistema ha detectado la activación del timbre de la propiedad.

Figura 6.13. El sistema detecta la activación del timbre y realiza la llamada telefónica.

6.2.4. DETECCIÓN DE UNA LLAMADA DEL USUARIO.

Para detectar la llamada de un usuario, al teléfono celular del sistema se le programó un tono de aviso compuesto únicamente de un patrón repetitivo de señales DTMF representada por el dígito “5” del teclado telefónico. Dicho tono de aviso se programó únicamente para representar las llamadas entrantes provenientes de los números telefónicos pertenecientes a los usuarios registrados y autorizados en el sistema, de manera que las llamadas provenientes de números no autorizados no poseen un tono de aviso sonoro y son rechazadas de forma automática sin que el sistema interactúe de forma insegura con usuarios desconocidos.

Una vez que el sistema recibe un patrón de señales DTMF del dígito “5” detectadas de manera consecutiva en 3 ocasiones, entonces envía de forma automática un comando AT vía Bluetooth al teléfono celular que ordene contestar a la llamada. Una vez que la llamada comienza, el usuario puede activar o desactivar periféricos de salida tales como la chapa electrónica con la tecla “4”, o el altavoz con la tecla “2”.

La figura 6.14 muestra la ventana principal de la aplicación mientras una llamada se está llevando a cabo. Nótese que el campo titulado “Contraseña” se agregó para confirmar que la detección de la llamada entrante se ha llevado a cabo correctamente y que el usuario ha decidido activar la chapa electrónica.

Figura 6.14. La llamada entrante se detecta, se responde y el usuario activa la chapa electrónica.

6.2.5. ACTIVACIÓN DEL ALTAVOZ.

Para que la activación del altavoz tenga sentido dentro de nuestro sistema de seguridad, primero deben efectuarse algunos eventos y acciones previas. Un ejemplo llevado a la práctica se describe a continuación.

Primero alguna persona activa el timbre de la propiedad, el sistema detecta este evento y procede a intentar establecer una llamada telefónica al usuario, éste recibe la llamada del sistema y responde a la misma, presionando además el dígito “1” de su teléfono para informar al sistema que la llamada se ha establecido y que no debe colgar o terminar la misma, bajo la probabilidad de que entre al buzón de voz ó simplemente que el usuario no contestó en un intervalo determinado. Después de que haya sucedido esto, finalmente el usuario activa el altavoz para comunicarse con la persona que se encuentre fuera de la propiedad.

La figura 6.15 muestra la ventana principal de la aplicación mientras la llamada se está llevando a cabo y el usuario activa el altavoz de la propiedad.

Figura 6.15. El altavoz (intercomunicador) es activado por el usuario, después de responder la llamada telefónica.

6.3. PRUEBAS CON RUIDO ACÚSTICO.

Bajo la posibilidad de que el sistema notifique al usuario de algún evento de interés mientras este se encuentre en un entorno de alto ruido acústico, resultó necesario llevar a cabo algunas pruebas bajo dicha situación y determinar si el sistema puede seguir trabajando con normalidad y más concretamente si se continúa realizando la detección de señales DTMF de manera satisfactoria.

A continuación se muestran las gráficas 6.16 y 6.17 de las DFT obtenidas por el sistema correspondientes a la señal DTMF de la tecla "1", éstas pueden ser comparadas entre sí para notar de manera visual las diferencias en los niveles de ruido y el rango de frecuencias en el que se presenta.

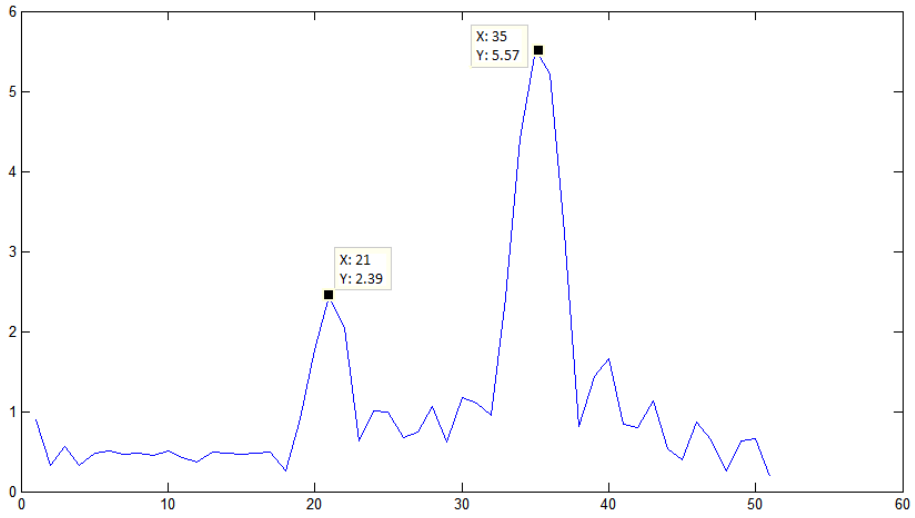


Figura 6.16. Gráfica de la señal de la tecla 1 en el dominio de la frecuencia y en condiciones de alto ruido acústico.

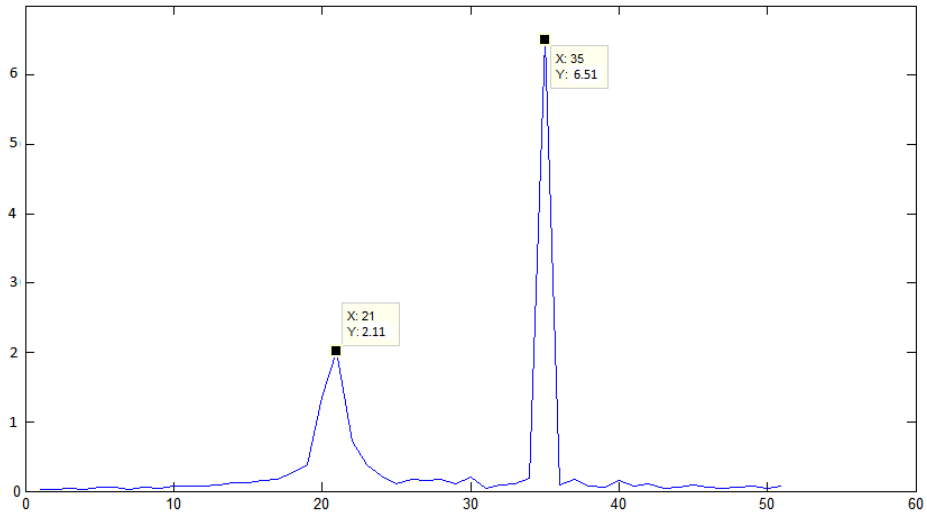


Figura 6.17. Gráfica de la señal de la tecla 1 en condiciones de muy escaso ruido acústico.

6.4. TABLA DE PRUEBAS Y RESULTADOS.

La siguiente tabla pretende mostrar un resumen básico y en algunos casos estadístico de los resultados encontrados al efectuar diversas pruebas al sistema.

PRUEBA	PORCENTAJE DE EFECTIVIDAD	RESULTADO U OBSERVACIONES
Detección de señales DTMF con bajo ruido ambiental	93%	Porcentaje promedio. Varía dependiendo de la tecla presionada. 10 muestras por tecla.
Detección de señales DTMF con alto ruido ambiental	89%	Porcentaje promedio. Varía dependiendo de la tecla presionada. 10 muestras por tecla.
Correcto reconocimiento de la tecla presionada con bajo ruido ambiental.	91%	En pocas ocasiones el sistema detecta la tecla o señal DTMF equivocada. Porcentaje promedio. 10 muestras por tecla.
Correcto reconocimiento de la tecla presionada con bajo ruido ambiental.	85%	Situación en la que más comprometido se ve el funcionamiento del sistema, sin embargo se obtiene un resultado aceptable.
Distancia máxima entre el teléfono celular local del sistema y la PC.	N/A	El enlace Bluetooth funcionó correctamente con hasta 11.5 metros de distancia aproximadamente en un espacio abierto.
Distancia máxima entre el microcontrolador y la PC.	N/A	Se probó con un cable USB de 8 metros y el sistema no presentó problemas de comunicación.
Distancia máxima entre el microcontrolador y el teléfono celular.	N/A	Menos de 3 metros con un cable sencillo de audio. Línea propensa a ruido e interferencia eléctrica por lo que entre menos es su longitud mejor resulta el funcionamiento.
Detección de apertura de puerta.	100%	N/A
Detección de apertura de ventana.	100%	N/A
Detección de apertura de activación de timbre.	100%	N/A
Detección de llamada entrante por parte del usuario.	80%	En ocasiones el tono creado con señales DTMF no fue detectado correctamente. Se realizaron 20 pruebas continuas

Tabla 6.1. Resumen de algunas pruebas y los resultados obtenidos.

6.5. RESUMEN Y SÍNTESIS.

La dificultad de mostrar la efectividad en los resultados obtenidos en las diversas pruebas efectuadas al sistema de seguridad es considerable, dado que en un trabajo escrito es muy complejo lograr plasmar los eventos y las acciones que se presentan en todo momento durante el funcionamiento del sistema.

Sin embargo, en el capítulo actual se pretendió presentar las imágenes de la aplicación o interfaz de usuario programada en la PC, creada con ciertas características que permitieran mostrar los eventos y acciones que se están llevando a cabo, como por ejemplo el campo de dígitos DTMF nombrado como “Contraseña” y de igual manera los bordes rectangulares que cambian de color cuando se presenta el evento que representan, en lugar de mostrar fotos de los periféricos que podrían no demostrar que los procesos se están realizando satisfactoriamente. Por ejemplo, una foto no puede probar que el altavoz se encuentra funcionando, que la chapa trabaja correctamente o que el teléfono responde a los comandos AT satisfactoriamente.

El procesamiento de las señales DTMF es una cuestión aparte. Las gráficas mostradas confirman que los principios aplicados del procesamiento digital de señales al tratamiento de las mismas corresponden de manera considerable a los conceptos expuestos en la teoría. Por lo que siendo este proceso el corazón del sistema de seguridad, el cual permite la comunicación y el flujo de información y comandos desde el sistema al usuario y viceversa, o en pocas palabras, siendo estas una especie de lenguaje que utiliza el usuario para comunicarse con el sistema, se pueden considerar como bastante exitosos los resultados obtenidos.

Por último se mostró brevemente la diferencia general que se presenta entre las gráficas de las señales DTMF en el dominio de la frecuencia, tanto cuando el sistema se encuentra trabajando en un entorno de alto ruido acústico como en uno de muy bajo ruido acústico. Cabe mencionar que el sistema no dejó de funcionar de manera aceptable aún cuando se estresó al máximo en dicho entorno de alto ruido.

CONCLUSIONES.

Se diseñó y construyó un sistema de seguridad capaz de detectar, comunicar y tomar acciones alrededor de diversos eventos que se puedan llevar a cabo en una casa o propiedad. Así mismo el sistema tiene la posibilidad de interactuar vía telefónica con el usuario para controlar diversas funcionalidades, respuestas y acciones. Todo esto en tiempo real.

Los algoritmos implementados son relativamente sencillos y no representan la parte teórica más compleja que puede llegar a tratarse en los estudios de procesamiento digital. Parte también del objetivo de este proyecto es demostrar que todos los conceptos estudiados pueden llevarse a cabo en aplicaciones reales y en diferentes áreas de la vida cotidiana.

Una instalación formal del prototipo de sistema de seguridad tiene que llevar todo un sistema de cableado de sensores y dispositivos, lo que conlleva naturalmente a que el sistema sea más propenso a interferencia eléctrica producida por la instalación eléctrica del mismo inmueble. La adición de algunos elementos adicionales, a los descritos en este texto, tendrían que ser requeridos tales como filtros, comparadores, amplificadores operacionales, cables de tipo coaxial o muchos otros dispositivos que pueden ayudar a anular los efectos de las señales externas al sistema.

Se detalló el efecto en las señales DTMF y su proceso de detección, cuando el usuario se encuentra en un entorno de elevado ruido acústico. Se mostraron gráficas que comprueban que aún en entornos muy extremos de ruido, el sistema emprendería una detección aceptable de señales DTMF y por ende presentaría un buen funcionamiento.

La distancia que puede haber entre la PC y el microcontrolador dependerá del cable USB utilizado, sin embargo la distancia máxima que puede haber entre el teléfono celular y la PC se encuentra limitada por la conexión Bluetooth de ambos dispositivos. Los teléfonos móviles

transmiten con una potencia que permite realizar enlaces con hasta aproximadamente 10 metros de distancia entre dispositivos.

Se describen pruebas y los resultados obtenidos para los principales procesos que se llevan a cabo en el sistema. Los más significativos son el alto porcentaje de efectividad en la detección de señales DTMF y su correcto reconocimiento. Como consecuencia de esto el sistema ejerce un número muy bajo de errores al reconocer los tonos correspondientes los dígitos del teclado telefónico.

Se pretende mostrar que el corazón del proyecto radica en el procesamiento digital de señales independientemente de la aplicación. Para este caso particular, la teoría tiene el objeto de fungir como un medio por el cual un usuario pueda interactuar con un sistema digital (de seguridad doméstica en este caso) por medio de las señales DTMF enviadas a través de una llamada telefónica.

Resulta necesario mencionar que no se pretende que el proyecto termine con este trabajo, sino que se busca seguir desarrollándolo para llevarlo hacia un estado ideal de funcionamiento. Ya sea implementando dispositivos más avanzados y algoritmos más eficientes, o bien implementando interfaces de comunicación aún más sencillas para los usuarios como puede ser por ejemplo, la interacción con el sistema mediante una llamada telefónica, pero esta vez utilizando comandos directos de voz.

Para ello es necesario implementar componentes de mayor capacidad, tales como microcontroladores de alta velocidad de procesamiento, DSP´s, PLD´s o FPGA´s , tal vez optimizar el trabajo del teléfono celular, mediante alguna aplicación diseñada para dispositivos móviles en algún lenguaje de programación, para que ello permitiera llevar a cabo el procesamiento del audio en el sistema, o simplemente trabajar más en poder profundizar en los diversos algoritmos de procesamiento y comunicación que hicieran aún más veloz y eficiente el sistema. Esta última opción depende a final de cuentas de un trabajo continuo que se pretende seguir ejerciendo, en la búsqueda no sólo de un sistema de seguridad más elegante, sino también en un aprendizaje más amplio en la rama del procesamiento digital de señales y su aplicación en los diversos proyectos de ingeniería.

REFERENCIAS Y BIBLIOGRAFÍA.

[1] Axelson J. *USB Complete: Everything You Need to Develop USB Peripherals, Third Edition*. Lakeview Research LLC. Madison WI. E.U.A. 2005.

[2] Escobar S. L. *Conceptos Básicos de PDS*. Facultad de Ingeniería, UNAM. México D.F. 2009.

[3] García B. E. *Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC*. Alfaomega. México D.F. 2008.

[4] García J., Rodríguez J. I. y Brazález A. *Aprenda Visual Basic 6.0 como si estuviera en primero*. Escuela Superior de Ingenieros Industriales, Universidad de Navarra. San Sebastián, España. 1999.

[5] Kammer D., McNutt G., Senese B. & Bray J. *Bluetooth Application Developer's Guide*. Syngress Publishing, Inc. Rockland MA, E.U.A. 2002.

[6] Rathi S. *Bluetooth Protocol Architecture.* Dedicated Systems Magazine, Microware Systems Corporation. 2000.

[7] The Mathworks, Inc. *MATLAB Help Index.* The Mathworks, Inc. E.U.A. 2009.

[8] Nokia Corp. *AT Command Set for Nokia GSM and WCDMA Products v1.2*, Nokia Corporation. E.U.A. 2005.

[9] Microchip, Inc. *PIC18F4550 Datasheet*. Microchip Technology Inc. E.U.A. 2009.

[10] Peñuelas R. U. M. Descripción de tarjeta: UPRsys V0308. Facultad de Ingeniería, UNAM. México D.F. 2008.

GLOSARIO.

A

A/D – Analógico-Digital.
ACL – Conexión Asíncrona (Bluetooth).
AM – Modulación en amplitud.
ASCII – Acrónimo de *American Standard Code for Information Interchange*.
ASK – Modulación por saltos en amplitud (*Amplitud Shift Keying*).
AT – Comandos AT. Abreviatura del vocablo inglés *attention*.

B

Bluetooth – Interfaz de comunicación inalámbrica.

C

CCS – Acrónimo de *Custom Computer Services* (Desarrollador del lenguaje PIC-C)
CDC – Clase de Dispositivo de Comunicación.

D

DCE – Acrónimo de *Data Circuit-Terminating Equipment* (Interfaz RS-232).
DFT – Transformada Discreta de Fourier.
DSP – Procesador de señales digitales (*Digital Signals Processor*).
DTMF – Tonos Duales de Multi-Frecuencia. Señales compuestas por dos sinusoidales simples (tonos) que oscilan en un rango de frecuencias desde 700 a 1600 Hz aproximadamente.
DTE – Acrónimo de *Data Terminal Equipment* (Interfaz RS-232).

F

FM – Frecuencia Modulada.
FSK – Modulación por saltos en frecuencia (*Frecuency Shift Keying)*.

G

GND – Tierra, potencial eléctrico de referencia en los diagramas eléctricos (*Ground*).
GFSK – Modulación Gausiana por saltos en frecuencia (*Gaussian Frecuency Shift Keying*).

H

HCI – Interface de Control del Huesped (*Host Controller Interface*).
HiD – Dispositivo de Interface Humana (*Human Interface Device*).

I

Ir – Infrarrojo.

L

LMP – Protocolo Gestor de Enlaces (*Link Manager Protocol*).
LSB – Bit menos significativo (*Less Significant Bit*).
L2CAP – Protocolo de enlace lógico de adaptación y control (*Logical Link Control and Adaptation Protocol*).

M

MSB – Bit más significativo (*Most Significant Bit*).

N

N/A – No Aplica.

O

OBEX – Protocolo de Intercambio de Objetos (*Object Exchange*).

P

PC – Computadora Personal.
PIC – Circuito Integrado Programable. Familia de microcontroladores diseñados y construidos por la compañía Microchip Inc.
PM – Modulación en fase (Phase Modulation).
PPP – Protocolo Punto a Punto (*Point to Point Protocol*).
PSK – Modulación por saltos en fase (*Phase Shift Keying*).

PWM – Modulación por ancho de pulso (*Pulse Width Modulation*).

Q

QAM – Modulación en cuadratura de amplitud.
QAPM – Modulación en cuadratura combinada de amplitud y fase.
QPM – Modulación en cuadratura de fase.

R

RD – Dato Recibido.
RFCOMM – Protocolo de comunicación serie vía Bluetooth.
RS-232 – Estándar recomendado número 232 (*Recommended Standard number 232*).

S

SCO/SCL – Conexión Síncrona (Bluetooth).

T

TD – Dato Transmitido.
TCS – Protocolo de Especificaciones de Control de Telefonía (*Telephony Control Specification*).
TCP/IP – Protocolo de Control de Transmisión e Internet (*Transmition Control Protocol* / *Internet Protocol*).

U

uC – Microcontrolador.
USART – Transmisor y Receptor, Síncrono y Asíncrono, Universal.
USB – *Universal Serial Bus*. Estandar de comunicación serie alámbrica.
USB-CDC – Tipo de archivo de biblioteca compuesta para emular un puerto serie RS-232 utilizando un puerto USB ordinario.

ANEXOS.

CÓDIGO PROGRAMADO EN LA PC CON LENGUAJE VISUAL BASIC 6.0.

```
Dim Num As String
Dim Mem As String
Dim s, m As String
Dim x, minutos As Integer

Dim Estado As Integer
Dim cosa1 As String
Dim k, conteo As Long
Dim n As Long
Dim Xr As Double
Dim Xi As Double
Dim Espec(250) As Double
Dim Tecla As String
Dim Pass As String
Dim Frec1 As Double
Dim Frec2 As Double

Dim Result As Long
Dim errormsg As Integer
Dim cadena As String * 1024
Dim CadenaError As String * 1024
Dim mssg As String * 255
Dim i As Long
Dim BlockAlign As Integer
Dim sBytes As String
Dim temp As String
Dim temp1 As Long

Dim lBytes As Long
Dim Datos(500) As Double

Private Sub cmdConts_Click()
   MSComm1.Output = "ATA" & vbCr
End Sub

Private Sub cmdMarcar_Click()
   Nombre = InStr(1, Mem, Combo1.Text)
```

```
   If Nombre = 0 Then
      tempX = MsgBox("¡El usuario a llamar no se encuentra registrado!", 0 + 48, "LLAMADA CANCELADA")
      GoTo 10:
   End If
   Num = Mid(Mem, Nombre, 1)
   Do While (Asc(Num) < 48 Or Asc(Num) > 57)
      Nombre = Nombre - (-1)
      Num = Mid(Mem, Nombre, 1)
   Loop
   Num = Mid(Mem, Nombre, 10)
   comando = "ATDT "
   fin_comando = ";"
   MSComm1.Output = comando + Num + fin_comando & vbCr
   'Text1.Text = Num
10:
End Sub

Private Sub cmdRedial_Click()
   MSComm1.Output = "AT+BLDN" & vbCr
End Sub

Private Sub cmdTecla_Click()
Dim m As Long

   Timer1.Enabled = False

   For j = 0 To 500
      temp = MSCommVirtual.Input
      MSCommVirtual.Output = "N" & vbCr
   Next j
   MSCommVirtual.Output = "K" & vbCr

   For j = 0 To 499
      temp = MSCommVirtual.Input
      If temp = "" Then
         temp = "€"
      End If
      temp1 = Asc(temp)
      Datos(j) = 2 * ((temp1 / 127) - 1)
      MSCommVirtual.Output = "N" & vbCr
   Next j
   MSCommVirtual.Output = "K" & vbCr

   'Timer1.Enabled = True

   For k = 0 To 250
```

```
      Xr = 0
      Xi = 0
      For n = 0 To 499
         Xr = Xr - (-(Datos(n) * Cos(2 * 3.1415926536 * k * n / 500)))
         Xi = Xi - (-(Datos(n) * Sin(2 * 3.1415926536 * k * n / 500)))
      Next n
      Espec(k) = Sqr((Xr ^ 2) - (-(Xi ^ 2)))
   Next k

   Frec2 = 1
   For k = 1 To 50
      If Espec(k) > Espec(Frec2) Then
         Frec2 = k
         'GoTo 10
      End If
   Next k

'10:
   Frec1 = 1
   For k = 1 To (Frec2 - 1)
      If Espec(k) > Espec(Frec1) Then
         Frec1 = k
         'GoTo 20
      End If
   Next k

'20:
   If Frec1 = 20 And Frec2 = 34 Then
      Tecla = "1"
   End If
   If Frec1 = 20 And Frec2 = 38 Then
      Tecla = "2"
   End If
   If Frec1 = 20 And Frec2 = 42 Then
      Tecla = "3"
   End If
   If Frec1 = 22 And Frec2 = 34 Then
      Tecla = "4"
   End If
   If Frec1 = 22 And Frec2 = 38 Then
      Tecla = "5"
   End If
   If Frec1 = 22 And Frec2 = 42 Then
      Tecla = "6"
   End If
   If Frec1 = 24 And Frec2 = 34 Then
```

```
    Tecla = "7"
  End If
  If Frec1 = 24 And Frec2 = 38 Then
    Tecla = "8"
  End If
  If Frec1 = 24 And Frec2 = 42 Then
    Tecla = "9"
  End If
  If Frec1 = 27 And Frec2 = 34 Then
    Tecla = "*"
  End If
  If Frec1 = 27 And Frec2 = 38 Then
    Tecla = "0"
  End If
  If Frec1 = 27 And Frec2 = 42 Then
    Tecla = "#"
  End If
  'If Frec1 = 20 And Frec2 = 18 Then
    'Text3.Text = ""
    'GoTo 30
  'End If
  Pass = Text1.Text + Tecla
  Text1.Text = Pass
  'Text1.Text = Frec1
30:
  Timer1.Enabled = True
  If Pass = "1" And Estado = 1 Then
    Timer3.Enabled = False
    Estado = 2
    Form2.Hide
  End If
  If Tecla = "2" And Estado = 2 Then
    MSCommVirtual.Output = "I" & vbCr
    'Estado = 3
  End If
  If Tecla = "3" And Estado = 2 Then
    'MSCommVirtual.Output = "I" & vbCr
    cmdColgar = True
    'Estado = 9
    'Timer3.Enabled = True
    'Form2.Hide
    'Estado = 0
  End If
  'Else
    'MSCommVirtual.Output = "N" & vbCr
  'End If
```

```
   Tecla = ""
End Sub

Private Sub cmdColgar_Click()

   MSComm1.Output = "AT+CHUP" & vbCr

End Sub

Private Sub Command1_Click()
   'Beep
   'mensaje = MsgBox("Mensaje de prueba", 0 + 0 + 64, "Prueba")
   'Shape6.BorderColor = &HFF00&
   'Form2.Caption = "APERTURA DETECTADA"
   'Form2.Show
   'MSComm1.Output = "ATDT 33352213;" & vbCr
   'h = " "
   'For j = 0 To 50
      'c = Str(Espec(j))
      'h = h + "  " + c
   'Next j
   'Text2.Text = h
End Sub

Private Sub cmdNuevo_Click()
   Form3.Show
End Sub

Private Sub Command2_Click()
   'tempX = MsgBox("¡Se eliminarán todos los usuarios registrados!", 1 + 48, "AVISO DE REGISTRO")
   'If tempX = 1 Then
      'Open "c:\Users\ALEX BUN\Documents\Predet.inf" For Output As #2
      'Print #2, "",
      'Close #2
      'Close #1
      'Open "c:\Users\ALEX BUN\Documents\Usuarios.inf" For Output As #1
      'Print #1, "",
      'Close #1
      'Open "c:\Users\ALEX BUN\Documents\Usuarios.inf" For Input As #1
   'Else

   'End If
End Sub

Private Sub Form_Activate()
```

```
    Combo1.Clear
    Mem = Input(LOF(1), #1)
    temp1 = 1
    Do
       temp2 = Mid(Mem, temp1, 14)
       Combo1.AddItem (temp2)
       temp1 = temp1 + 28
    Loop While ((temp1 < LOF(1)) Or (temp1 = LOF(1)))

    Close #1
    Open "c:\Users\ALEX BUN\Documents\Predet.inf" For Input As #2
    Mem = Input(14, #2)
    temp2 = Mid(Mem, 1, 14)
    Combo1.Text = temp2
    Close #2
    Open "c:\Users\ALEX BUN\Documents\Usuarios.inf" For Input As #1
    Mem = Input(LOF(1), #1)

End Sub

Private Sub Form_Deactivate()
    Open "c:\Users\ALEX BUN\Documents\Predet.inf" For Output As #2
    Print #2, Combo1.Text
    Close #2
    'Shape6.BorderColor = &HC0&
End Sub

Private Sub Form_Load()

    'Configuración de los puertos de comunicación.
    MSComm1.CommPort = 10
    MSComm1.Settings = "128000,N,8,1"
    MSComm1.PortOpen = True
    MSCommVirtual.CommPort = 9
    MSCommVirtual.Settings = "128000,N,8,1"
    MSCommVirtual.InputMode = comInputModeText
    MSCommVirtual.PortOpen = True

    'Abre la base de datos de los usuarios del sistema.
    Open "c:\Users\ALEX BUN\Documents\Usuarios.inf" For Append As #1
    Close #1
    Open "c:\Users\ALEX BUN\Documents\Usuarios.inf" For Input As #1
    Open "c:\Users\ALEX BUN\Documents\Predet.inf" For Append As #2
    Print #2, "",
    Close #2
```

```
   'Inicializa algunos estados del sistema.
   conteo = 0
   Estado = 0
   Form2.Enabled = False

End Sub

Private Sub Form_Unload(Cancel As Integer)
   Open "c:\Users\ALEX BUN\Documents\Predet.inf" For Output As #2
   Print #2, Combo1.Text
   Close #2
End Sub

Private Sub Timer1_Timer()
   Label5.Caption = Time
   cosa1 = MSCommVirtual.Input
   If cosa1 = "" Then
         cosa1 = "€"
   End If
   cosa2 = Asc(cosa1)
   'Text1.Text = cosa2
   If cosa1 = "L" Then
      MSCommVirtual.Output = "K"
      cmdTecla = True
   End If
   If cosa2 = 129 And Estado = 0 Then
      'Shape1(0).BorderColor = &HFF00&
      Form2.Show
      cmdMarcar = True
      Estado = 1
      Timer3.Enabled = True
   End If
   'If Pass = "12" And Estado = 2 Then
   '    MSCommVirtual.Output = "I" & vbCr
      'Shape1(0).BorderColor = &HC0&
      'Shape5.BorderColor = &HFF00&
   '    Estado = 3
   'Else
   '    MSCommVirtual.Output = "N" & vbCr
   'End If
   If conteo > 20 Then
      conteo = conteo + 1
      If conteo > 32 Then
         conteo = 0
      End If
   End If
```

```
    MSCommVirtual.Output = "N" & vbCr
End Sub

Private Sub Timer2_Timer()
    Label5.Caption = Time
    Clave1 = MSCommVirtual.Input
    Clave2 = Asc(Clave1)
    Text3.Text = Clave2
    If Clave2 = "B" And conteo = 0 Then
        Form2.Show
        cmdMarcar = True
        Estado = 1
        Timer2.Enabled = False
        Timer3.Enabled = True
        Timer1.Enabled = True
    End If
    If conteo > 20 Then
        conteo = conteo + 1
        If conteo > 30 Then
            conteo = 0
        End If
    End If
End Sub

Private Sub Timer3_Timer()
    Label5.Caption = Time
    'Text3.Text = conteo
    conteo = conteo + 1
    If conteo > 20 Then
        Timer3.Enabled = False
        cmdColgar = True
        Form2.Hide
        Estado = 0
    End If
    If Estado = 9 Then
        Timer3.Enabled = False
        cmdColgar = True
        Form2.Hide
        Estado = 0
    End If
End Sub
```

CÓDIGO PROGRAMADO EN EL MICROCONTROLADOR EN LENGUAJE CCS PIC C.

```
#include <18F4550.h>
#device ADC=8
#fuses HSPLL,NOWDT,NOPROTECT,LVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=20000000)
#use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7)

#define USB_HID_DEVICE     FALSE          //deshabilitamos el uso de las directivas HID
#include "usb_cdc.h"

void main(void)
{
char comp,resp,sensor,datos[500];
int16 cont;
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(0);
delay_us(10);
usb_init();
usb_cdc_init();
usb_task();
output_low(PIN_C0);
output_low(PIN_C1);
output_low(PIN_C2);
do
{
  output_high(PIN_C0);
  delay_ms(300);
  output_low(PIN_C0);
  delay_ms(300);
}
while(!(usb_enumerated()==TRUE));
output_high(PIN_C0);
comp=127;
sensor=0;
  while(true)
  {
    sensor=INPUT_D();
    comp=read_adc();
    if ((comp>135)||(comp<120))
    {
      output_low(PIN_C0);
      cont=0;
      do
      {
```

```
        datos[cont]=read_adc();
        cont++;
        delay_us(100);
      }
      while(cont<500);
      do
      {
        usb_cdc_putc('L');
        do
        {
          comp=usb_cdc_getc();
        }
        while(usb_cdc_kbhit()==FALSE);
      }
      while(!(comp=='K'));
      cont=0;
      do
      {
        usb_cdc_putc(datos[cont]);
        cont++;
        do
        {
          comp=usb_cdc_getc();
        }
        while(usb_cdc_kbhit()==FALSE);
      }
      while((!(comp=='K'))||(cont<=500));
      output_high(PIN_C0);
    }
    else
    {
      output_high(PIN_C0);
      usb_cdc_putc_fast(sensor);
      do
      {
        resp=usb_cdc_getc();
      }
      while(usb_cdc_kbhit()==FALSE);
      if (resp=='I')
      {
        output_toggle(PIN_C1);
        output_toggle(PIN_C2);
      }
    }
  }
}
```