UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE TECNOLOGÍA

CARRERA ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE GRADO
NIVEL LICENCIATURA

SISTEMA DE SUPERVISIÓN REMOTA PARA DOMICILIOS U OFICINAS MEDIANTE LA RED GSM Y GPRS

Postulante: Henry Alcon Choque
Tutor: Lic. Nixon Emiliano Vargas Mamani

La Paz – Bolivia
2016

AGRADECIMIENTOS.

Quiero agradecer especialmente a mis padres por su paciencia, comprensión y por el apoyo que me brindaron.

También agradecer a mi tutor de proyecto, quien con su ayuda y su guía fue parte importante para la conclusión satisfactoria de este proyecto.

A mis amigos un agradecimiento especial ya que gracias a ellos pude superarme constantemente.

ÍNDICE GENERAL.

ÍNDICE DE FIGURAS

RESUMEN.

El presente proyecto se basa en la necesidad de contar con un sistema de seguridad para domicilios u oficinas que brinde un medio de notificación al dueño y a la empresa a cargo de la seguridad cuando suceda la intrusión del delincuente.

Con este proyecto se pretende lograr disuadir al delincuente en su intención de cometer el robo, brindar al dueño del domicilio mayor seguridad con respecto a los bienes que posee y lograr que las empresas de seguridad y vigilancia tengan un medio óptimo para desempeñar su trabajo.

Para lograr este objetivo se implementará un servidor que recibirá los mensajes de alerta generados por los equipos de monitoreo que serán implementados con capacidad de conexión a internet mediante la red GPRS y en su defecto mediante la red GSM vía SMS.

CAPITULO I

FUNDAMENTOS ESPECÍFICOS DEL PROYECTO.

1.1. INTRODUCCIÓN.

Desde siempre se ha tenido el problema de la seguridad en los domicilios, saber que está pasando, el olvido de la llave para la puerta de ingreso, el apagado de las luces cuando se sale por la noche, en el caso de seguridad el problema es mayor ya que cuando un domicilio se encuentra sin habitantes es propenso a sufrir robos.

Debido a los peligros que representan los domicilios deshabitados se creara un sistema seguridad y supervisión remota que sea accesible para toda persona no importando el lugar donde se encuentre su domicilio, esto siempre y cuando se cuente con el servicio de energía eléctrica y exista un operador de telefonía móvil (GSM) que tenga cobertura sobre la zona donde se encuentre el domicilio.

1.2. ANTECEDENTES.

La delincuencia ha sido siempre motivo de alerta para las personas, ya que estos delincuentes cometen delitos como asaltos, otra forma de operar de los delincuentes es el robo en domicilios que se encuentran deshabitados periódicamente ya que los propietarios por diversos motivos tienen que ausentarse, esto no solo pasa en los domicilios particulares sino también en locales comerciales donde los dueños una vez terminada la jornada de trabajo deben retirarse a sus domicilios dejando solo el local, lo mismo pasan con las oficinas.

A causa de estos eventos se crean empresas de seguridad y vigilancia, para lo cual se contrata personal que hacen la función de “serenos”, estos tienen la función de vigilar las calles mediante rondas periódicas o constantes, pero no pueden estar en todos los lugares a la vez, por tanto existen periodos en los que los domicilios, locales comerciales y oficinas están desprotegidos pudiendo ser aprovechados estos momentos por delincuentes para cometer el robo.

Debido a estas situaciones es necesario tener un sistema que sea capaz de monitorear el estado de los domicilios, locales comerciales y oficinas en todo momento.

1.3. PLANTEAMIENTO DEL PROBLEMA.

La proliferación de los robos a locales comerciales, domicilios, oficinas, etc. Hacen necesario que los propietarios tomen acciones de seguridad, en los casos más comunes, candados chapas o aldabas y otros, resulta que estos medios ya no son confiables, ya que se dan casos en los que los delincuentes cortan partes de la puerta (puerta metálica) para así burlar los candados que de sitúan por dentro de la puerta para luego cometer el robo, en otros casos también sucede a que los malhechores son capases de perforar la pared para acceder al negocio o domicilio y así saquear todo.

Debido a estos extremos en el actuar de los delincuentes, es necesario contar con un sistema de alarma con sensores que detecten movimientos, ruptura de vidrios, etc., además de mantener informado al dueño de lo que sucede en su local o domicilio.

1.4. OBJETIVO GENERAL.

- Diseñar un sistema de supervisión domiciliaria remota mediante la red GSM/GPRS,

1.5. OBJETIVOS ESPECÍFICO.

- Implementar una base de datos que contenga toda la información necesaria para para el funcionamiento del sistema.
- Crear un software de monitoreo para que los supervisores puedan conocer el estado de eventos en el domicilio monitoreado.
- Implementar un servidor web para que el dueño del domicilio pueda tener acceso a la información generada por el equipo de monitoreo.
- Crear un protocolo de tramas de datos para intercambiar información entre el servidor soket y el equipo de monitoreo.
- Implementar un servidor socket para recibir información de los eventos generados por los equipos de monitoreo.
- Diseñar e implementar el equipo de monitoreo que se encargara de efectuar la lectura de los sensores ubicados en el domicilio y enviar esa información hacia el servidor socket mediante la red GPRS y en su defecto mediante la red GSM en forma de SMS.

- Crear el firmware del microcontrolador que gestionara todas las tareas del equipo de monitoreo.
- Crear un software de configuración del equipo der monitoreo mediante puerto USB.

1.6. JUSTIFICACIÓN TÉCNICA.

En la actualidad el control de la seguridad está gobernada por dispositivos de marcado automático mediante línea telefónica (esto en empresas que cuentan con tal servicio), para lo cual es imprescindible contar con línea telefónica fija, de modo que si el delincuente logra cortar la línea el local queda sin comunicación (hasta que la empresa de seguridad note la falta de comunicación con el local).

También es evidente que en nuestro país no todas las personas cuentan con línea telefónica fija.

Con un sistema inalámbrico como el GSM este problema está prácticamente eliminado, por tanto es más seguro

1.7. JUSTIFICACIÓN SOCIAL.

Las sociedades en su conjunto siempre ha tenido el problema de la inseguridad debido a delincuentes que buscan adueñarse de lo ajeno. Una sociedad con altos índices de delincuencia y robos a domicilios y locales comerciales es una sociedad insegura, esa falta de seguridad repercute en el progreso mismo de la sociedad, si los locales comerciales son objeto de robos en momentos en que se encuentra cerrado o en ausencia del propietario es probable que cierre definitivamente por causa de la pérdida del capital o en otro caso la reducción del personal para mitigar las perdidas y esto genera un menor movimiento económico.

En el caso de un domicilio si a este se le roba los bienes inmuebles es probable que el propietario no adquiera unos nuevos o costosos por temor a ser robado otra vez o en otro caso para recuperar lo perdido el propietario reduce su gasto habitual afectando el comportamiento económico del entorno.

Un sistema de supervisión remota en un local comercial o domicilio es capaz de generar alertas sonoras y de aviso a los propietarios, a la empresa a cargo de la seguridad y mediante los anteriores a las autoridades competentes y esto contribuye la sociedad sea más segura.

Una zona o localidad que tenga instalado sistemas de seguridad y monitoreo tiene una menor probabilidad de que el delincuente trate de hacer laguna fechoría ya que pone en peligro su libertad.

1.8. JUSTIFICACIÓN ECONÓMICA.

Un sistema de monitoreo domiciliario puede presentar un problema a la hora de decidir si instalar el equipamiento o no. Si bien el costo es una preocupación válida, se deberá tomar en consideración los riesgos que puede llegar a correr.

Proteger a la familia y las posesiones son importante para la mayoría de las personas. Un sistema de monitoreo domiciliario disminuye el riesgo de tener a un ladrón entrando a robar. Los criminales evitan se alejarán de las casas donde haya algún tipo de mecanismo de seguridad. La mayoría de los ladrones viajan por el barrio buscando las casas que sean más fáciles de asaltar, y las eligen antes que aquellas que tengan sistemas de seguridad que puedan poner su libertad el riesgo, Incluso si llegan a ingresar al domicilio se irán rápidamente (tomando menos cosas y minimizando el daño) cuando suene una alarma.

Utilizar un sistema de seguridad provee seguridad a la persona, su familia y sus posesiones. El estar a salvo de los robos es invaluable.

La implementación del Sistema de monitoreo domiciliario implicará una inversión pero a cambio el mismo permitirá evitar posteriores robos que derivan en muchos casos con pérdidas cuantiosas.

1.9. LIMITES.

EL presente proyecto está limitado a áreas del país donde no exista cobertura de la red GSM/GPRS ya que usa como base de comunicación para sus reportes el sistema telefonía móvil a través de un modem GSM/GPRS.

1.10. ALCANCES.

Este proyecto plantea ser una alternativa a los sistemas de vigilancia comerciales existentes en el mercado nacional debido a que al tener toda la documentación respectiva se puede implementar más prestaciones o simplemente ser de propósito más específico.

El proyecto podrá extender su aplicación para todo tipo mando a distancia incluyendo domótica, telemetría, rastreo de vehículos, sistemas scada, etc.

CAPITULO II

FUNDAMENTACIÓN TEÓRICA.

2.1. SERVIDOR WEB CON XAMPP.

XAMPP, es un software de distribución libre, su nombre deriva del acrónimo **X** (para los distintos sistemas operativos) **A**pache **M**ySQL **P**HP **P**erl, está bajo licencia GNU GLP, es un servidor que puede interactuar con páginas dinámicas, XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOSX. Una de las ventajas de XAMPP es que de una forma muy sencilla y rápida se puedes levantar un servidor y por tanto entorno de desarrollo de cualquier aplicación web que use PHP y base de datos.

2.2. CONFIGURACIÓN DE XAMPP.

Una vez instalado el XAMPP se tendrá una ventana similar a esta (XAMPP 1.6.4).

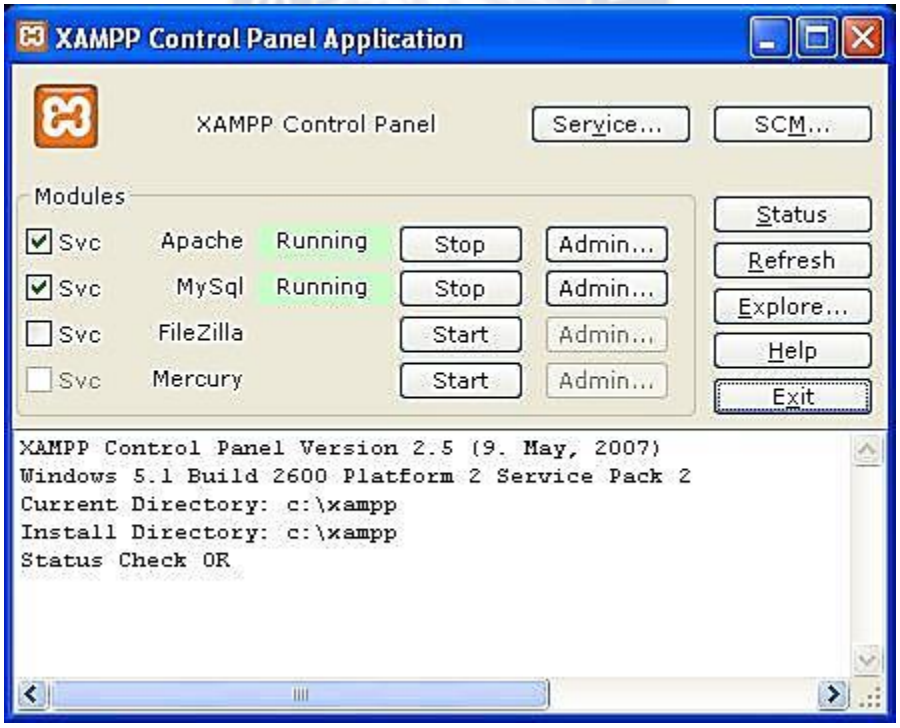


Figura 0.1 Panel de control del servidor XAMPP.

En esta ventana se puede iniciar o detener los servicios del servidor, para iniciar un servicio bastara con oprimir el botón *Start* y para detenerlo se oprimirá *Stop*.

Para hacer las configuraciones iníciales se deberá usar un navegador, por ejemplo y en este caso el *explorer* de Microsoft, en el navegador se escribe el siguiente enlace "**http://localhost**", y se muestra una página al cual hay que hacerle *click* en el idioma en el que se desea hacer la configuración.

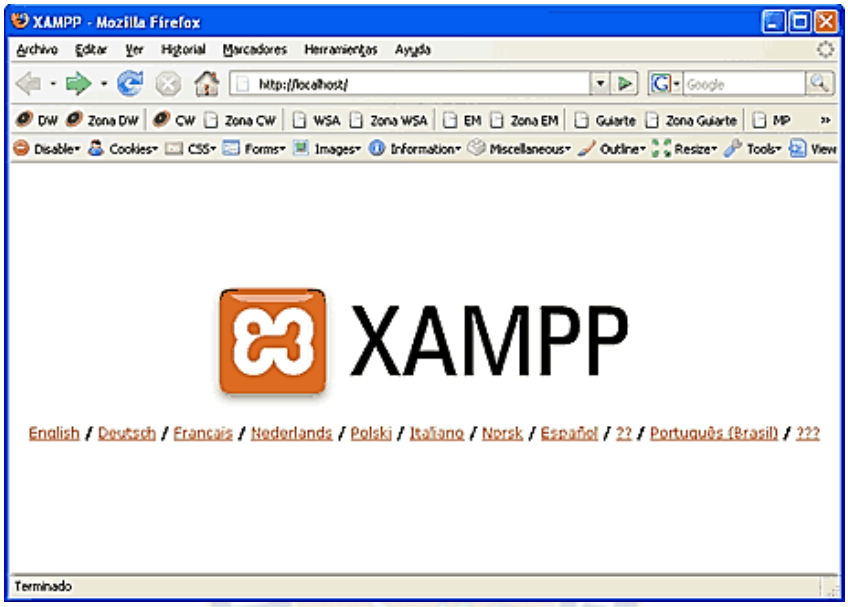


Figura 0.2 Página de selección de idioma la configuración del servidor.

Una vez seleccionado el idioma (Español), se tiene la siguiente ventana de configuraciones.

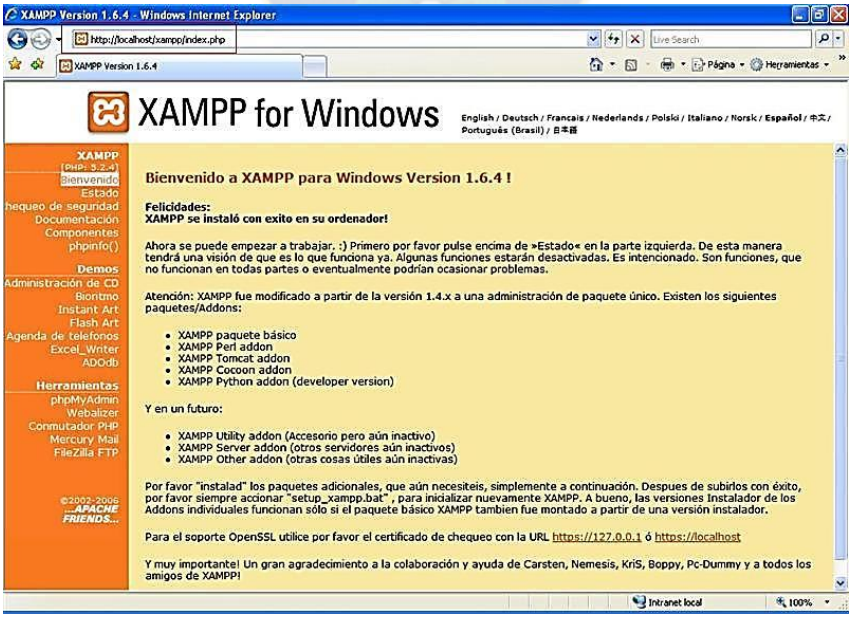


Figura 0.3 Ventana de configuraciones para el servidor.

Se observa que en la primera sección se encuentra toda la información referente al PHP que se instaló.

Para hacer las configuraciones de seguridad se debe ir al siguiente enlace "http://localhost/security/xamppsecurity.php".

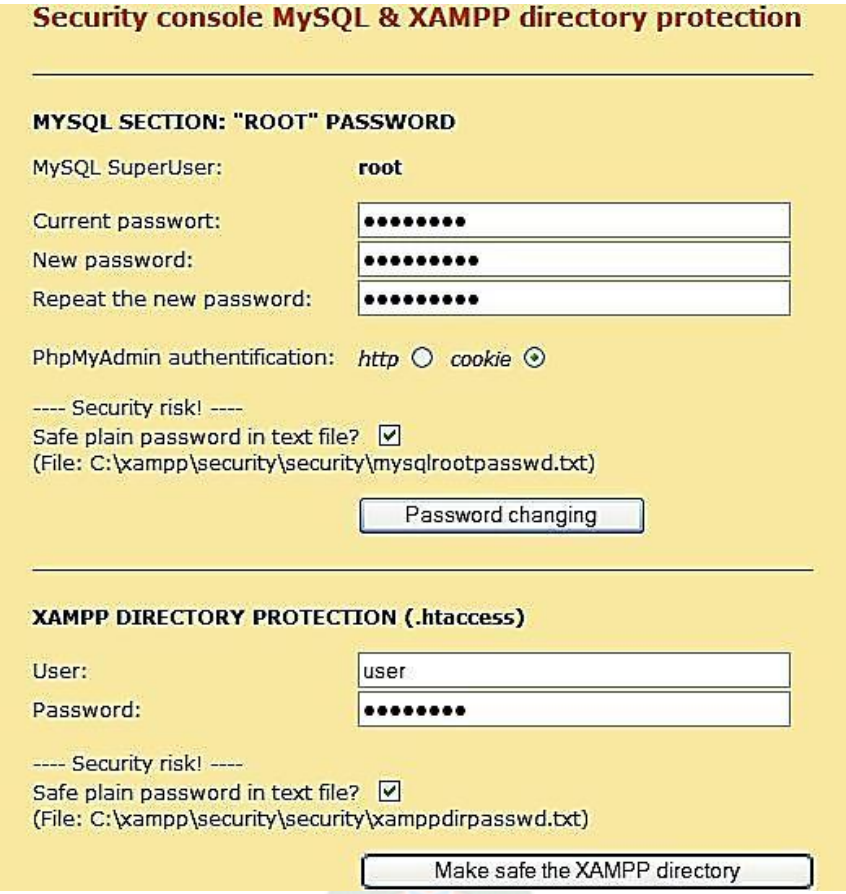


Figura 0.4 Ventana de configuración de seguridad para el servidor y la base de datos.

Antes de iniciar las configuraciones en esta ventana se debe hacer algunos cambios.

Inicialmente MySQL crea un usuario por defecto llamado root sin password. Para cambiar el password de root se debe acceder a la administración de MySQL a través del Panel de Control de XAMPP. Después de guardar los cambios, hay que modificar el fichero "**config.inc.php**" situado en "…HOME_XAMPP/phpMyAdmin/" y editar las siguientes líneas:

```
$cfg['Servers'][$i]['auth_type']     = 'config';
$cfg['Servers'][$i]['user']          = 'root';
$cfg['Servers'][$i]['password']      = 'XXXXXX'
```

En este fichero se configurarán las variables necesarias para que phpMyAdmin pueda acceder a MySQL, las más importantes son auth_type para el tipo de autenticación, user y password. Para la variable auth_type podemos poner el método de autenticación **http** y cuando accedamos a phpMyAdmin nos aparecerá una ventana para introducir el usuario y password de MySQL. Sin embargo, si ponemos como método de autenticación **config** debemos poner en las variables user y password el usuario y password de MySQL y de esta forma accederá directamente a phpMyAdmin sin preguntar nada al usuario.

Figura 0.5 Ventana del administrador PHP.

La carpeta destinada a guardar las aplicaciones web es "**htdocs**" que está ubicada en la carpeta principal de XAMPP.

Figura 0.6 Carpeta en donde se alojan las aplicaciones web.

Para poder ver las aplicaciones creadas bastara con introducir en la barra de direcciones del navegador el path relativo a partir de la carpeta "htdocs" justo después de "http://localhost" (para este caso "**http://localhost/tutorial**").

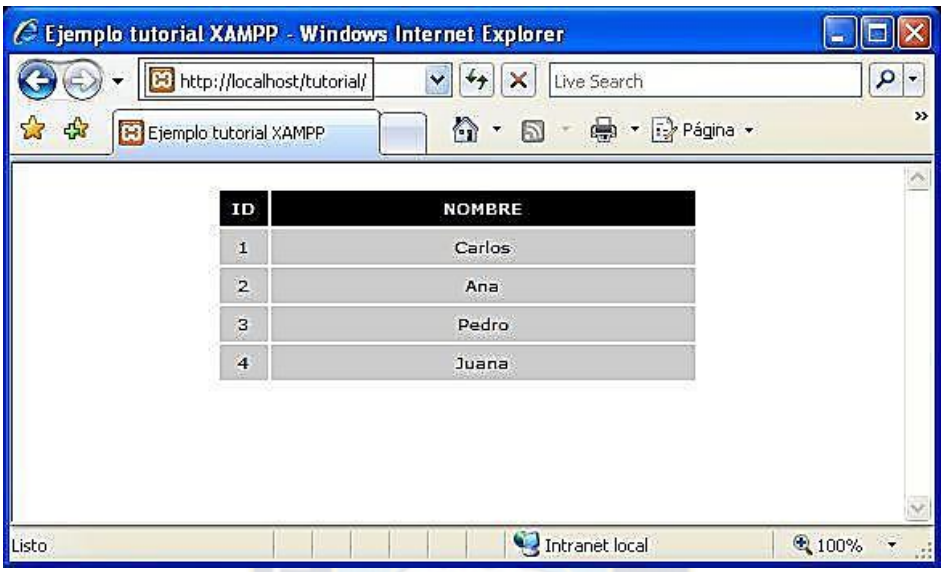


Figura 0.7 Resultado de la aplicación web.

2.3. EL HTML.

El HTML (Lenguaje de marcación de hipertexto) se genera a partir de dos conceptos, el concepto de Hipertexto y el SGML.

El Hipertexto es el medio por el cual se permite enlazar distintos documentos entre sí.

El SGML (Lenguaje estándar de marcación general) es el que permite colocar etiquetas o marcas en un texto para definir el formato de presentación del texto.

HTML no tiene ningún compilador y por ello si existiera algún error de sintaxis este no será detectado y se visualizara en el modo como el intérprete lo entienda.

En Windows el intérprete de HTML puede ser cualquier navegador como ser el Explorer, Firefox, etc.

2.3.1. ESTRUCTURA DE UN DOCUMENTO HTML.

Un documento HTML está constituido por tres partes principales.

- Inicio y fin del documento. Inicia con <html>, y termina con </html>

- Encabezamiento, se delimita con las etiquetas <head> y </head>
- Cuerpo del documento, se delimita con las etiquetas <body> y </body>

La etiqueta <html> indica al cliente (computadora que accede al servidor) que comienza un documento HTML.

La cabecera <head>, aquí se pone información sobre el documento, esta información no se ve en la pantalla del navegador. Aquí se puede introducir un título <title> y </title> y es lo que se verá como título en la parte superior del navegador.

El cuerpo del documento <body>, aquí es donde se introducirá la información que se desea mostrar en la pantalla del navegador.

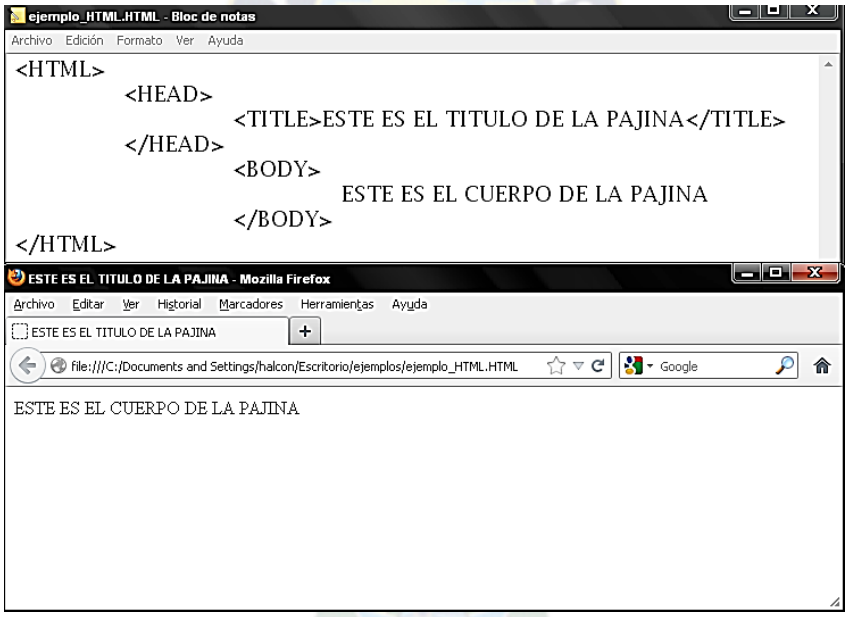


Figura 0.8 Ejemplo de la estructura básica de un documento HTML.

2.3.2. COMANDOS BÁSICOS DE FORMATO DE TEXTO.

- Comienzo de párrafo … , cuenta con opciones de alineado, align="center", "left" o "right", como es para el inicio de un párrafo, este comando deja un espacio separador.
- Tamaño de letra <h1>…</h1>, como en un documento de texto, este comando permite definir el tamaño de las letras, puede varias de *h1* hasta *h7*.
- Poner al centro un texto … .

- Letras de paso fijo, estas letras se caracterizan por ser tosas del mismo tamaño <pre>…</pre>
- Retorno de carro, en HTML el retorno de carro no tiene efecto, para lograr el retorno de carro se usa
.
- Línea horizontal o separador <hr>, se pueden modificar algunos atributos. Grosor de la barra horizontal size=”num”, ancho de la barra horizontal width=”num”, alineamiento align=”…”, puede tener parámetros como *left*, *right* y *center*, *noshade*, elimina el aspecto tridimensional.

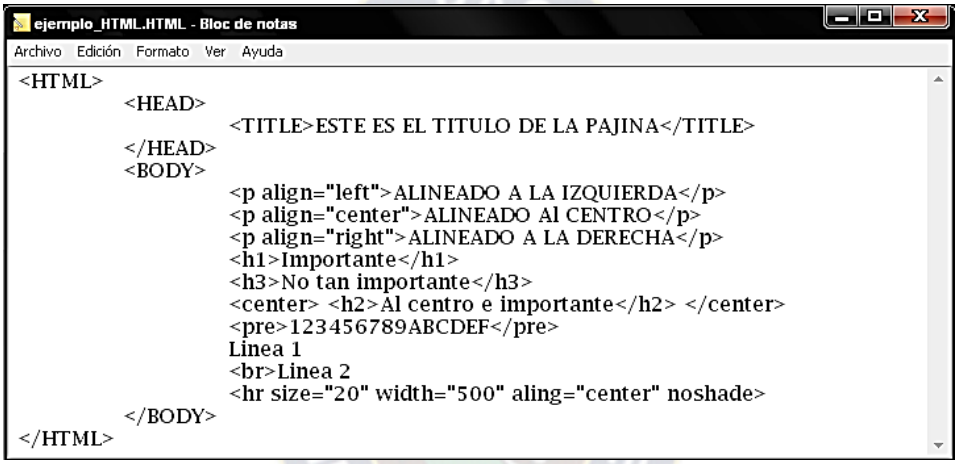


Figura 0.9 Ejemplo básico del uso de comandos de formato de texto en HTML.

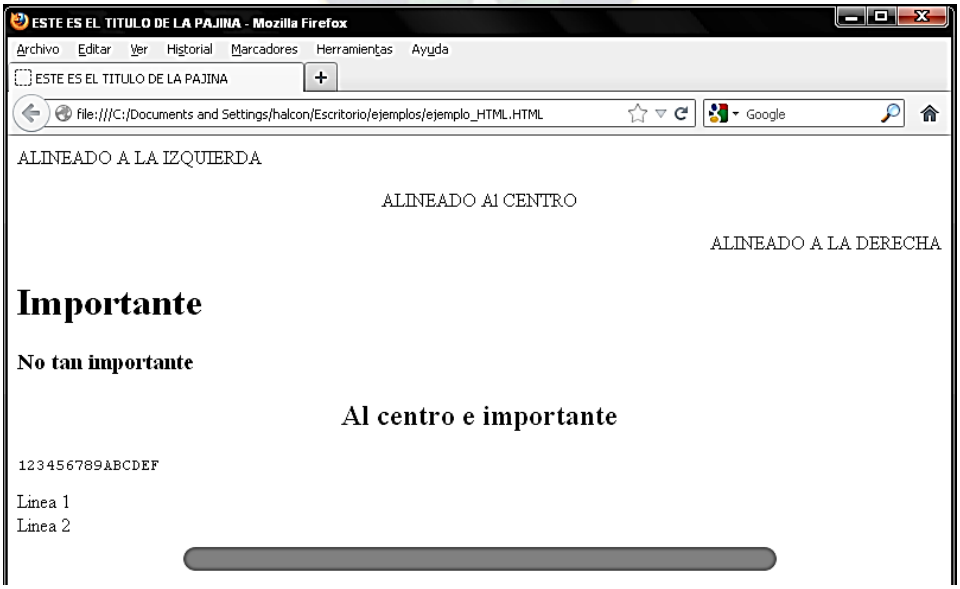


Figura 0.10 Resultado del ejemplo básico del uso de comandos de formato de texto en HTML.

2.3.3. CARACTERES ESPECIALES EN HTML.

Al elaborar una página HTML con un editor de texto se puede utilizar tildes, caracteres que son propios del castellano (á, Á, É, ü, Ñ,¿,¡) y el navegador presentara de manera adecuada estos caracteres ya que por defecto asume que la página está escrita en caracteres del idioma europeo occidental. En otras regiones del mundo los navegadores no están configurados el mismo modo, para notificar a otros navegadores que estamos utilizando caracteres del idioma europeo occidental, la primera línea de la cabecera deberá ser.

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

2.3.4. ENLACES.

Un enlace es una zona de texto o grafico que nos traslada a otra posición del documento actual o a otro documento, este documento puede encontrarse en el mismo ordenador o en otro que puede estar en cualquier parte del planeta.

Para generar un enlace se utiliza la etiqueta <a>…, todo lo que se encierra dentro de esta etiqueta es considerado como enlace, este enlace sufrirá dos modificaciones.

1. El texto aparecerá subrayado con un color distinto al habitual y las imágenes estarán rodeados por un borde dl mismo color que el texto de enlace.
2. Al pulsar el enlace, se abrirá en el navegador el documento al que apunta el enlace.

En un enlace hay que especificar la dirección y se lo hace de la siguiente manera.

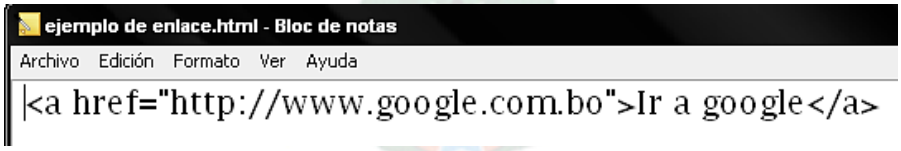


Figura 0.11 Ejemplo de un enlace en HTML.

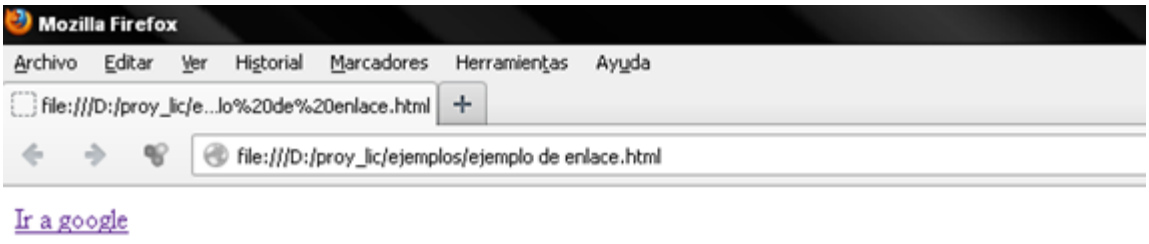


Figura 0.12 Resultado del ejemplo de enlace en HTML.

2.3.5. INCLUIR IMÁGENES EN HTML.

Para incluir gráficos e imágenes en las páginas se utiliza la etiqueta <img>.

<img src="fichero_grafico" alt="descripción">

El parámetro **src** especifica el nombre del fichero que contiene el gráfico. Los formatos estándar en la red son el GIF, JPG y PNG.

El parámetro **alt** especifica el texto que se mostrará en lugar del gráfico en aquellos navegadores que no sean capaces de mostrarlos o en el supuesto de que el usuario los haya desactivado.

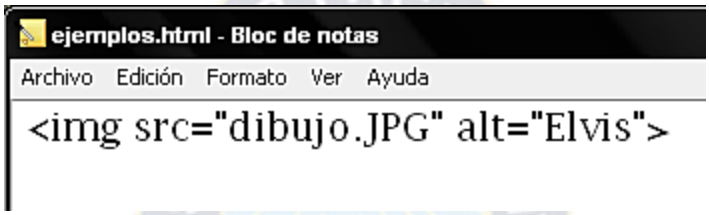


Figura 0.13 Ejemplo de imagen en HTML.

Figura 0.14 Resultado del ejemplo de imagen en HTML.

2.3.6. TABLAS EN HTML.

Para generar una tabla se usa las etiquetas <table>… </table>.

Cada fila de la tabla está definida por <tr> y </tr>

Cada celda está definida por <td> y </td>

Las etiquetas <th> y </th> sirven para indicar que el texto contenido es la cabecera de la columna. Las cabeceras de la tabla no son obligatorias.

La etiquetas <caption> y </caption> delimitan el título de la tabla.

Los atributos de <table> son:

border="num", especifica el grosor del borde que se dibujará alrededor de las celdas. Por defecto es cero, lo que significa que no dibujará borde alguno.

cellspacing="num", define el número de pixels que separarán las celdas.

cellpadding="num", especifica el número de pixels que habrá entre el borde de una celda y su contenido.

width="num", "%", especifica la anchura de la tabla. Puede estar tanto en pixels como en porcentaje de la anchura total disponible para ella (pondremos "100%" si queremos que ocupe todo el ancho de la ventana del navegador).

align="...", alinea la tabla a izquierda ("left"), derecha ("right") o centro ("center").

bgcolor="#xxxxxx", colorea todo el área de la tabla con el color indicado.

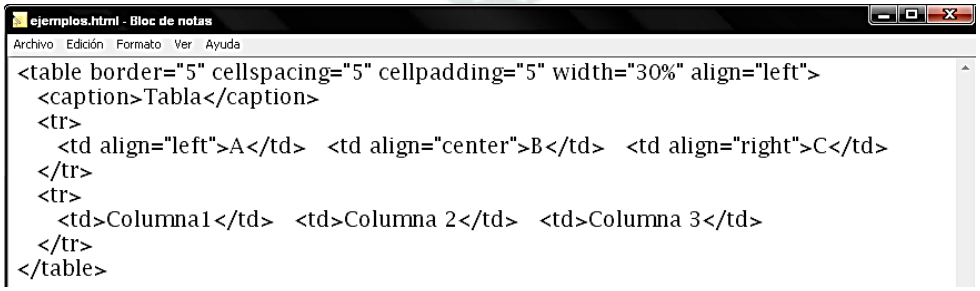


Figura 0.15 Ejemplo de Tablas en HTML.

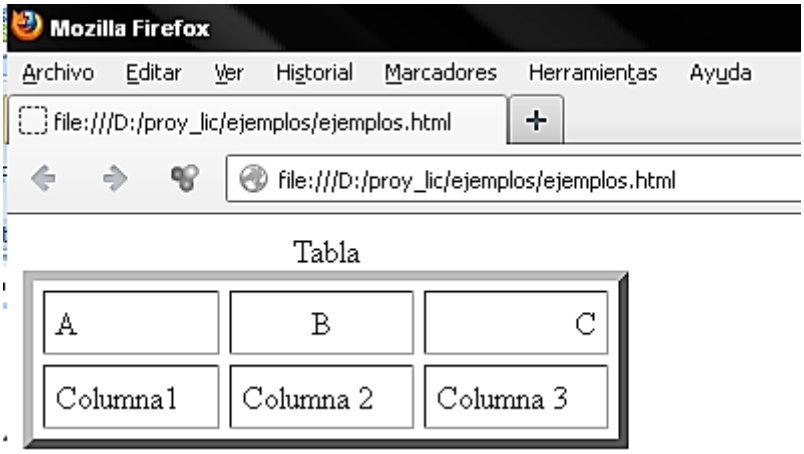


Figura 0.16 Resultado del ejemplo de tablas en HTML.

2.4. PHP.

PHP (Hypertext Pre-processor) es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

2.4.1. ETIQUETAS DE PHP.

Cuando PHP interpreta un fichero, busca las etiquetas de apertura y cierre, que son <?php y ?>, y que indican a PHP dónde empezar y finalizar la interpretación del código. Este mecanismo permite a PHP ser incrustado en todo tipo de documentos, ya que todo lo que esté fuera de las etiquetas de PHP será ignorado por el intérprete.

PHP también permite las etiquetas abreviadas <? y ?>

2.4.2. PHP Y HTML.

Cualquier cosa fuera del par de etiquetas de apertura y cierre es ignorado por el intérprete de PHP, el cual permite que los ficheros de PHP tengan contenido mixto. Esto permite que PHP sea embebido en documentos HTML para, por ejemplo, crear plantillas.

Esto va a ser ignorado por PHP y mostrado por el navegador.

<?php echo 'Mientras que esto va a ser interpretado.';?>

Esto también será ignorado por PHP y mostrado por el navegador.

2.4.3. COMENTARIOS.

PHP soporta comentarios 'C', 'C++'

```
<?php
   echo 'Esto es una prueba'; // Esto es un comentario estilo c++ de una sola línea
   /* Esto es un comentario multilínea
      y otra línea de comentarios */
   echo 'Esto es otra prueba';
?>
```

2.4.4. VARIABLES.

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado seguido de cualquier número de letras, números y caracteres de subrayado.

```
<?php
$var = “Roberto”;
$Var = “Juan”;
echo "$var, $Var";     // imprime "Roberto, Juan"
$4site = “aun no”;       // inválido; comienza con un número
$_4site = “aun no”;     // válido; comienza con un carácter de subrayado
$täyte = “mansikka”;    // válido; 'ä' es ASCII (Extendido) 228
?>
```

PHP también ofrece otra forma de asignar valores a las variables: asignar por referencia. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" o "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa.

Para asignar por referencia, simplemente se antepone un signo ampersand (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente segmento de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = “Bob”;              // Asigna el valor 'Bob' a $foo
$bar = &$foo;              // Referenciar $foo vía $bar.
$bar = "Mi nombre es $bar";  // Modifica $bar...
echo $bar;
echo $foo;                 // $foo también se modifica
?>
```

Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

```
<?php
$foo = 25;
$bar = &$foo;      // Esta es una asignación válida.
$bar = &(24 * 7);  // Inválida; referencia una expresión sin nombre.
function test()
{
  return 25;
}
$bar = &test();    // Inválido.
?>
```

2.4.4.1.ÁMBITO DE UNA VARIABLE

El ámbito de una variable es el contexto dentro del que la variable está definida. La mayor parte de las variables PHP sólo tienen un ámbito simple. Este ámbito simple también abarca los ficheros incluidos y los requeridos. Por ejemplo:

```
<?php
$a = 1;
include “b.inc”;
?>
```

Aquí, la variable $a estará disponible al interior del script incluido *b.inc*. Sin embargo, al interior de las funciones definidas por el usuario se introduce un ámbito local a la función. Cualquier variable usada dentro de una función está, por omisión, limitada al ámbito local de la función. Por ejemplo:

```
<?php
$a = 1; /* ámbito global */
function test()
{
   echo $a; /* referencia a una variable del ámbito local */
}
test();
?>
```

Este script no producirá salida, ya que la sentencia *echo* utiliza una versión local de la variable $a, a la que no se ha asignado ningún valor en su ámbito. Puede que usted note que hay una pequeña diferencia con el lenguaje C, en el que las variables globales están disponibles automáticamente dentro de la función a menos que sean expresamente sobrescritas por una definición local. Esto puede causar algunos problemas, ya que la gente puede cambiar variables globales inadvertidamente. En PHP, las variables globales deben ser declaradas globales dentro de la función si van a ser utilizadas dentro de dicha función.

```
<?php
$a = 1;
$b = 2;
function Suma()
{
   global $a, $b;
   $b = $a + $b;
}
Suma();
echo $b;
?>
```

El script anterior producirá la salida 3. Al declarar $a y $b globales dentro de la función, todas las referencias a tales variables se referirán a la versión global. No hay límite al número de variables globales que se pueden manipular dentro de una función.

2.4.4.2.VARIABLES DESDE FUENTES EXTERNAS

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP. Por ejemplo, consideremos el siguiente formulario:

```
<form action="foo.php" method="post">
   Nombre usuario: <input type="text" name="username" /><br />
   Email:  <input type="text" name="email" /><br />
   <input type="submit" name="submit" value="¡Enviarme!" />
</form>
```

Para acceder a los datos de formularios HTML

```
<?php
  $usuario= $_POST['username'];
  $correo= $_POST['email'];
?>
```

2.4.5. CONSTANTES.

Una constante es un identificador (nombre) para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script. (A excepción de las constantes predefinidas, que en realidad no son constantes). Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores de contantes siempre suelen declararse en mayúsculas.

SINTAXIS.

Se puede definir una constante usando la función define().Una vez que la constante está definida, no puede ser cambiada o redefinida en ningún momento.

Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se debe prefijar una constante con el signo $. También se puede usar la función constant() para leer el valor de una constante si se desea obtener el valor de una constante de forma dinámica. Use get_defined_constants() para obtener una lista de todas las constantes definidas.

Estas son las diferencias entre constantes y variables:

- Las constantes no llevan el signo dólar ($) como prefijo.
- Las constantes solo pueden ser definidas usando la función define (), y no por simple asignación.
- Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.
- Las constantes no pueden ser redefinidas o eliminadas una vez se han definido. Y
- Las constantes solo deberían contener valores escalares.

```
<?php
define("CONSTANT", "Hola mundo.");
echo CONSTANT; // muestra "Hola mundo."
echo Constant; // muestra "Constant" y provoca un error.
?>
```

2.4.6. OPERADORES.

Los operadores se pueden agrupar de acuerdo con el número de valores que toman. Los operadores unarios toman sólo un valor, por ejemplo ! (el operador lógico de negación) o ++ (el operador de incremento). Los operadores binarios toman dos valores, como los familiares operadores aritméticos + (suma) y - (resta), y la mayoría de los operadores de PHP entran en esta categoría. Finalmente, hay sólo un operador ternario, ? :, El cual toma tres valores; usualmente a este se le refiere simplemente como "el operador ternario" (aunque podría tal vez llamarse más correctamente como el operador condicional).

2.4.6.1.OPERADORES ARITMÉTICOS.

Operadores aritméticos

Ejemplo	Nombre	Resultado
-$a	Negación	Opuesto de $a.
$a + $b	Adición	Suma de $a y $b.
$a - $b	Sustracción	Diferencia de $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de $a dividido por $b.

El operador de división ("/") devuelve un valor flotante a menos que los dos operando sean integers (o strings que se conviertan a integers) y los números sean divisibles, en cuyo caso será devuelto un valor integer.

Los operando del módulo se convierten en integers (por extracción de la parte decimal) antes del procesamiento.

2.4.6.2.OPERADORES BIT A BIT.

Operadores bit a bit

Ejemplo	Nombre	Resultado
$a & $b	And (y)	Los bits que están activos en ambos $a y $b son activados.
$a | $b	Or	Los bits que están activos ya sea en $a o en $b son activados.
$a ^ $b	Xor	Los bits que están activos en $a o en $b, pero no en ambos, son activados.
~ $a	Not (no)	Los bits que están activos en $a son desactivados, y viceversa.
$a << $b	Shift left	Desplaza los bits de $a, $b pasos a la izquierda (cada paso quiere decir "multiplicar por dos").
$a >> $b	Shift right	Desplaza los bits de $a, $b pasos a la derecha (cada paso quiere decir "dividir por dos").

El desplazamiento de bits en PHP es aritmético. Los bits desplazados por fuera de cualquiera de los extremos son descartados. Desplazamientos de izquierda tienen ceros desplazados a la derecha mientras que el bit de signo es desplazado fuera a la izquierda, es decir que no se conserva el signo de un operando. Desplazamientos a la derecha tienen copias del bit de signo desplazado a la izquierda, es decir que se conserva el signo de un operando.

Se debe utilizar para garantizar la precedencia deseada. Por ejemplo, $a & $b == true evalúa la equivalencia y luego el bit a bit, mientras que ($a & $b) == true evalúa el bit a bit y luego la equivalencia.

Si tanto los parámetros de la izquierda como los de la derecha son cadenas, el operador bit a bit operará en los valores ASCII de los caracteres.

2.4.6.3.OPERADORES LÓGICOS.

Operadores lógicos

Ejemplo	**Nombre**	**Resultado**
$a and $b	And	TRUE si tanto $a como $b son TRUE.
$a or $b	Or	TRUE si cualquiera de $a o $b es TRUE.
$a xor $b	Xor	TRUE si $a o $b es TRUE, pero no ambos.
! $a	Not	TRUE si $a no es TRUE.
$a && $b	And	TRUE si tanto $a como $b son TRUE.
$a || $b	Or	TRUE si cualquiera de $a o $b es TRUE.

2.4.7. FUNCIONES.

2.4.7.1.FUNCIONES DEFINIDAS POR EL USUARIO.

Una función puede ser definida usando una sintaxis como la siguiente:

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
   echo "Función de ejemplo.\n";
   return $valordevuelto;
}
?>
```

Los nombres de las funciones siguen las mismas reglas que otras etiquetas de PHP. Un nombre de función válido comienza con una letra o guion bajo, seguido de cualquier número de letras, números, o guiones bajos.

2.4.7.2.ARGUMENTOS DE FUNCIONES.

La información puede ser pasada a las funciones mediante la lista de argumentos, la cual es una lista de expresiones delimitadas por comas. Los argumentos son evaluados de izquierda a derecha.

```
<?php
function tomar_array($entrada)
{
   echo "$entrada[0] + $entrada[1] = ".$entrada[0]+$entrada[1];
}
?>
```

2.4.7.3.DEVOLVER VALORES.

Los valores son devueltos usando la sentencia opcional *return*. Se puede devolver cualquier tipo, incluidos arrays y objetos. Esto causa que la función finalice su ejecución inmediatamente y pase el control de nuevo a la línea desde la que fue llamada.

```
<?php
function cuadrado($núm)
{
   return $núm * $núm;
}
echo cuadrado(4);   // imprime '16'.
?>
```

2.4.7.4.ENVIANDO VARIABLES POR MÉTODO POST.

En PHP no hay una manera directa de enviar variables por el método POST, para tal efecto se puede hacer uso de una pequeña clase que permite el envío de parámetros POST de una manera sencilla.

```
require_once 'Request.php';
$request = new Request();
$request->addParam("nombre", "Pepe");
$request->addParam("apellido", "Mackoy");
// ...
$request->forward("destino.php");
```

El archivo Requiest.php es el siguiente (se hiso una copia literal del autor de este articulo).

```
<?php
//////////////////////////////////////////////////
// Request.php
//
```

```
// Esta clase permite intercambiar datos entre una
// Página y otra mediante GET y POST. Mediante esta
// Clase se suple la carencia que hay en PHP de hacer
// Un envío de datos por POST de manera natural.
//
// Puedes darle cualquier uso a esta clase siempre y
// Cuando seas respetuoso de los derechos de autoría
// y menciones el sitio de donde la obtuviste.
//
// http://www.ammeza.com
//
// Copyright (C) 2006 - 2010 Alejandro Morales Meza
//
// License: LGPL, see LICENSE
////////////////////////////////////////////////

define('GET', 'get');
define('POST', 'post');

/*
 * Permite enviar datos entre una página y otra utilizando
 * los métodos GET y POST.
 *
 * @author Alejandro Morales Meza
 * @copyright 2006 - 2010 Alejandro Morales Meza
 */
class Request {

    /*
     * Almacena un par [clave => valor] con cada uno de los
     * Parámetros a enviar.
     *
     * @var array
     */
          private $params = NULL;

    /**
     * Indica si los datos se van a enviar por GET o POST.
     * Su valor por defecto es 'post'
     *
     * @var string
     */
          private $method = NULL;

    /**
     * Representa el atributo target de la etiqueta <form>
     * e indica hacia que frame se enviar? los datos. Su
```

```
 * valor por defecto es '_self'.
 *
 * @var <type>
 */
    private $target = NULL;

/**
 * Constructor de la clase.
 *
 * @param string $method
 * @param string $target
 */
    function Request($method = POST, $target = "_self") {
        $this->params = array();
        $this->method = $method;
        $this->target = $target;
    }

/**
 * Agrega un par [clave => valor] al objeto.
 * La clave no puede ser numérica.
 *
 * @param string $key
 * @param string $value
 */
    function addParam($key, $value) {
        if (is_numeric($key)) {
            throw new Exception("El nombre del parámetro no puede ser numérico.");
        }

        $this->params[$key] = $value;
    }

/**
 * Devuelve el valor que se encuentra almacenado en el
 * objeto bajo la clave pasada como paramento.
 *
 * @param string $key
 * @return string
 */
    function getParam($key) {
        return $this->params[$key];
    }

/**
 * Adiciona un conjunto de datos a los que ya se encuentran
 * en el objeto. Si una clave se repite, es reemplazada
```

```
     * con el nuevo valor.
     *
     * @param array $params
     */
        function setParams($params) {
                if (is_array($params)) {
                        foreach ($params as $key => $value) {
                                $this->addParam($key, $value);
                        }
                }
        }

    /*
     * Ejecuta el envío de parámetros a la página especificada
     * en el parámetro 'url'.
     *
     * @param string $url
     * @param bool $execute
     */
        function forward($url, $execute = true) {
                $max = sizeof($this->params);
                $str = "";
                foreach ($this->params as $key => $value) {
                        $str .= "<input name=\"{$key}\" type=\"hidden\" value=\"{$value}\">";
                }
                $html =
                        "<html>".
                        "<head>".
                        "<script>".
                        "function post_forward() {".
                        ($execute ? "document.getElementById(\"post_form\").submit();" : "").
                        "}".
                        "</script>".
                        "</head>".
                        "<body onload=\"post_forward()\">".
                        "<form id=\"post_form\" name=\"post_form\" method=\"{$this-
>method}\" action=\"$url\" target=\"{$this->target}\">".
                        "$str".
                        "</form>"."</body>"."</html>";
                print $html;
        }
}
?>
```

2.5. VISUAL BASIC 6.0.

Visual Basic 6.0 que para el caso de los programadores expertos facilita el desarrollo de aplicaciones complejas en poquísimo tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo). En el caso de los programadores novatos se convierte en una herramienta amigable para empezar su aprendizaje. El precio que hay que pagar por utilizar Visual Basic 6.0 es una menor velocidad o eficiencia en las aplicaciones.

Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4ª generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Visual Basic 6.0 es también un programa basado en objetos, aunque no orientado a objetos como C++ o Java. La diferencia está en que Visual Basic 6.0 utiliza objetos con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo propios de los verdaderos lenguajes orientados a objetos como Java y C++.

2.5.1. PROGRAMAS ORIENTADOS A EVENTOS.

Los programas orientados a eventos son los programas típicos de Windows, tales como Netscape, Word, Excel y PowerPoint. Cuando uno de estos programas ha arrancado, lo único que hace es quedarse a la espera de las acciones del usuario, que en este caso son llamadas eventos.

2.5.2. TIPOS DE PROYECTOS EN VISUAL BASIC.

Al iniciar el entorno de Visual Basic se observa se puede elegir el tipo de proyecto con el que se va a trabajar.

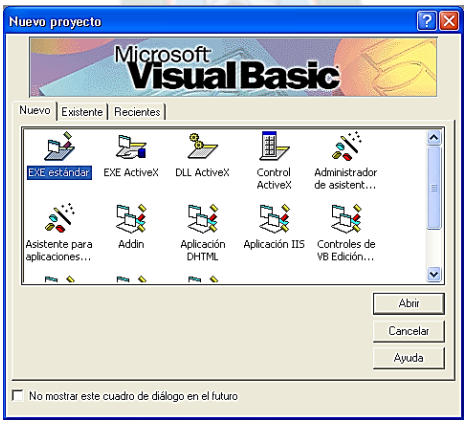


Figura 0.17 Ventana de Dialogo inicial en Visual Basic 6.

Asistente para aplicaciones de VB. Nos guía mediante una serie de pasos, en la construcción de una aplicación, creando la funcionalidad básica, ahorrándonos el trabajo más frecuente, de forma que sólo tengamos que incluir código en los puntos que deseemos personalizar.

Administrador de asistentes de VB. Permite configurar cada uno de los asistentes del entorno.

2.5.2.1.PROYECTO.

Aquí es donde se agrupan los diferentes módulos que contiene la aplicación: formularios, módulos de código, módulos de clase, etc., organizados en diferentes carpetas.

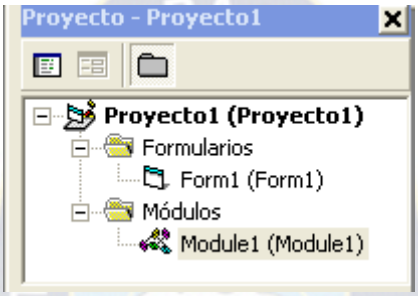


Figura 0.18 Ventana de Proyecto de una aplicación..

2.5.3. FORMULARIOS.

El formulario es el eje central en los programas, y se le pueden incluir los controles necesarios para que actúe de interfaz entre el usuario y la aplicación. Puesto que es un objeto, posee varias propiedades que pueden ser modificadas para adaptarlo a los requerimientos de la aplicación, así como métodos, en los cuales el programador puede insertar código para que se comporte según sea necesario.

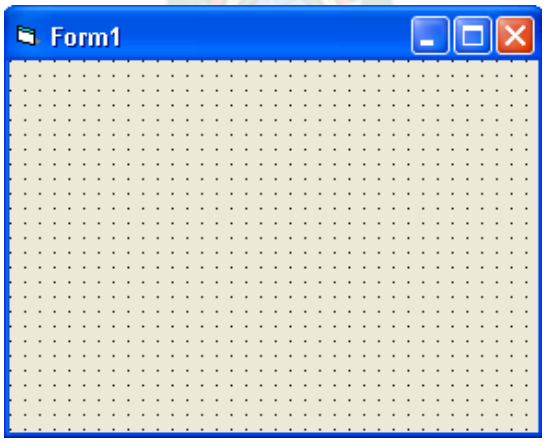


Figura 0.19 Formulario de un programa.

2.5.4. CONTROLES.

Son todos los elementos incluidos dentro de un formulario: botones de comando, listas desplegables, rejillas, cuadros de texto, botones de opción, de chequeo, menús, etc., en los cuales el usuario inserta información, selecciona valores o pulsa con el ratón para desencadenar ciertas acciones. Son objetos al igual que los formularios, por lo que también disponen de propiedades para cambiar su aspecto o posición, y métodos para responder a los eventos que reciban.

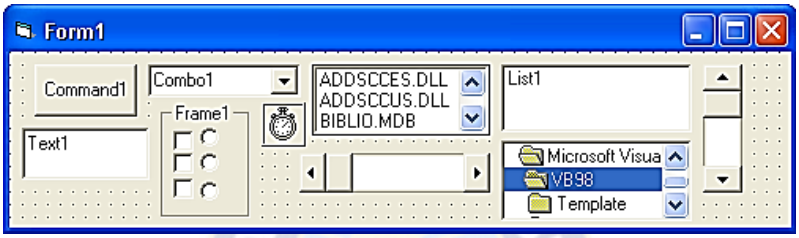


Figura 0.20 Formulario conteniendo varios controles.

2.5.5. PROPIEDADES.

Tanto los formularios como controles disponen de propiedades que el programador puede alterar. La manera de hacerlo es mediante la ventana de propiedades de VB, que mostrará las propiedades del objeto a modificar.

Dependiendo del objeto con el que estemos trabajando, esta ventana mostrará propiedades comunes para todos los objetos como Name, y otras propiedades que serán particulares para un tipo determinado de objetos, por ejemplo ControlBox para objetos formulario.

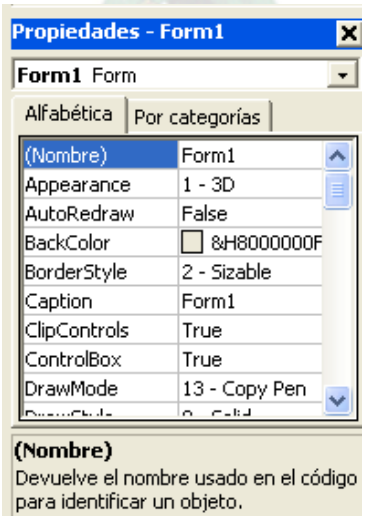


Figura 0.21 Ventana de propiedades del formulario.

2.5.6. EVENTOS Y MÉTODOS.

Se puede incluir código asociado a los objetos para que se comporten de la forma que se desea al recibir una acción del usuario. Para ello disponemos de la ventana de código del formulario, en la que situaremos las instrucciones que debe seguir un control al recibir un determinado evento.

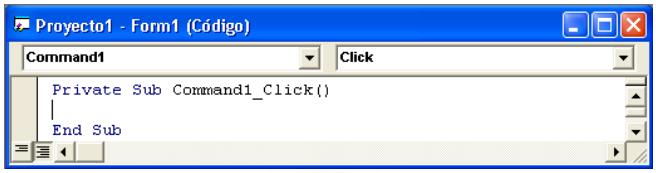


Figura 0.22 Ventana de código con el método vacío de pulsación de un botón.

2.5.7. SOCKETS EN VISUAL BASIC.

Los controles que Visual Basic tiene preparados para explotar los recursos de la red hacen deducir que existe uno específico para comunicarse con otro ordenador mediante TCP y UDP.

2.5.8. CONTROL WINSOCK.

El control WinSock permitirá conectarse a un equipo e intercambiar datos con el protocolo TCP. A partir de lo dicho, se podrá crear una aplicación que resida en un servidor y al que se pueda acceder desde varios clientes. Es lo que se conoce como una aplicación Cliente/Servidor.

El control Winsock es el más elemental de los controles que Visual Basic dedica a las redes IP, ya que lo único que hace en sí es efectuar la conexión.

No implementa ningún servicio, eso se tiene que hacer con un programa pero en contraposición nos permite tener muchos datos y mucho control sobre ella. Es decir, una vez establecida la conexión, se establecerá un servicio mediante código.

El control Winsock habrá que agregarlo en la caja de herramientas (Proyecto > Componentes >). Su nombre completo es Microsoft Winsock Control 6.0. No es visible en tiempo de ejecución, sólo el programador sabe que el control se encuentra en nuestra aplicación y cuáles son sus propiedades. Tiene esta forma:

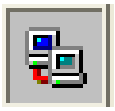


Figura 0.23 Icono del control WinSock.

2.5.8.1.PROPIEDADES

Property BytesReceived As Long (Solo lectura). Retorna el número de Bytes recibidos en la conexión.

Property Index As Integer (Solo lectura). Retorna/Asigna el número que identifica al control en un arreglo de controles.

Property LocalHostName As String (Solo lectura). Retorna el nombre de la maquina local.

Property LocalIP As String (Solo lectura). Retorna la dirección IP de la maquina local.

Property LocalPort As Long (Solo lectura). Retorna el puerto usado en la maquina local.

Property Protocol As ProtocolConstants. Retorna/Asigna el tipo de protocolo que usara el Socket, estos valores pueden ser dos: sckTCPProtocol y sckUDPProtocol.

Property RemoteHost As String. Retorna/Asigna el nombre (dirección) usado para identificar a la maquina remota.

Property RemoteHostIP As String (Solo lectura). Retorna la dirección IP del Host Remoto.

Property RemotePort As Long. Retorna/Asigna el puerto al cual se conectara en el computador remoto.

Property SocketHandle As Long (Solo lectura). Retorna el manejador (Handle) del Socket. (Solo para usuarios avanzados)

Property State As Integer (Solo lectura). Retorna el estado de la conexión del Socket.

2.5.8.2.MÉTODOS

Sub Accept(requestID As Long). Acepta una petición de conexión entrante.

Sub Bind([LocalPort], [LocalIP]). Amarra un socket a un específico puerto y adaptador.

Sub Close(). Cierra la conexión actual.

Sub Connect([RemoteHost], [RemotePort]). Conecta a un computador remoto.

Sub GetData(data, [type], [maxLen]). Recibe datos enviados por el computador remoto.

Sub Listen(). Se pone a la escucha de peticiones de conexión entrantes.

Sub PeekData(data, [type], [maxLen]). Mira los datos entrantes sin removerlos desde el buffer.

Sub SendData(data). Envía datos al computador remoto.

2.5.8.3.EVENTOS

Event Close(). Ocurre cuando la conexión a sido cerrada remotamente.

Event Connect(). Ocurre cuando la operación de conexión se ha completo.

Event ConnectionRequest(requestID As Long). Ocurre cuando un cliente remoto se intenta conectar.

Event DataArrival(bytesTotal As Long). Ocurre cuando se reciben datos desde un computador remoto.

Event Error(Number As Integer, Description As String, Scode As Long, Source As String, HelpFile As String, HelpContext As Long, CancelDisplay As Boolean). Se dispara cuando ocurre algún error.

Event SendComplete(). Ocurre después que una operación de envió se haya completado.

Event SendProgress(bytesSent As Long, bytesRemaining As Long. Ocurre mientras se está enviando un dato.

2.5.8.4.CONSTANTES

Enum StateConstants

Estas son las constantes devueltas por la propiedad **State**.

sckClosed. El socket se encuentra cerrado.

sckClosing. El socket está cerrando la conexión al computador remoto.

sckConnected. El socket ha conectado al computador remoto.

sckConnecting. El socket está conectando al computador remoto.

sckConnectionPending. El socket tiene una petición pendiente.

sckError. Se ha producido un error.

sckHostResolved. El socket ha resuelto el nombre del equipo remoto.

sckListening. El socket está a la escucha de peticiones entrantes.

sckOpen. El socket está actualmente abierto.

sckResolvingHost. El socket está resolviendo el nombre del computador remoto.

- **Enum ProtocolConstants.**

Estos son los valores permitidos por la propiedad **Protocol**.

sckTCPProtocol. Protocolo TCP.

sckUDPProtocol. Protocolo UDP.

- **Enum ErrorConstants.**

Constantes de errores, devueltas por el evento **Error.**

sckAddressInUse. Dirección en uso.

sckAddressNotAvailable. La dirección no está disponible desde la maquina local.

sckAlreadyComplete. La operación está completa. En progreso la operación de no bloqueo.

sckAlreadyConnected. El socket esta actualmente conectado.

sckBadState. Protocolo o estado de conexión equivocada para la transacción o petición demandada.

sckConnectAborted. La conexión es abortada debido a que se ha superado el tiempo de conexión u otra falla.

sckConnectionRefused. La conexión a sido rechazada.

sckConnectionReset. La conexión a sido reinicializada por el lado remoto.

sckGetNotSupported. La propiedad es de Escritura solamente.

sckHostNotFound. Respuesta autorizada. No se ha encontrado el Host.

sckHostNotFoundTryAgain. Respuesta no autorizada. No se ha encontrado el Host.

sckInProgress. Una operación winsock de bloqueo está en progreso.

sckInvalidArg. El argumento pasado a la función no posee un formato o rango valido.

sckInvalidArgument. Argumento invalido

sckInvalidOp. Operación invalida en el estado actual.

sckInvalidPropertyValue. Valor de propiedad invalido.

sckMsgTooBig. El datagrama es muy largo para acomodarlo dentro del buffer y ha sido truncado.

sckNetReset. Ha finalizado el tiempo de conexión cuando SO_KEEPALIVE estaba seteado.

sckNetworkSubsystemFailed. Falla en el subsistema de red.

sckNetworkUnreachable. La red no puede ser alcanzada desde este host.

sckNoBufferSpace. No hay espacio disponible en el buffer,

sckNoData. Nombre valido. No hay tipo de datos del registro demandado.

sckNonRecoverableError. Error irrecuperable.

sckNotConnected. El socket no está conectado.

sckNotInitialized. WinsockInit debe ser llamado primero.

sckNotSocket. El descriptor no es un socket.

sckOpCanceled. Esta operación es cancelada.

sckOutOfMemory. La memoria se ha colapsado.

sckOutOfRange. El argumento esta fuera de rango.

sckPortNotSupported. El puerto especificado no está soportado.

sckSetNotSupported. La propiedad es de solo lectura.

sckSocketShutdown. El socket había sido cerrado.

sckSucces. Concluido.

sckTimedout. Tiempo fuera en el intento de conexión.

sckUnsupported. No soporta tipos Variants.

sckWouldBlock. El socket no está bloqueando y la específica operación será realizada.

sckWrongProtocol. Protocolo equivocado para la específica transacción o petición.

2.5.8.5.APLICACIÓN CLIENTE.

Esta aplicación trabajara como un cliente simple que conecte a cualquier servidor, permita enviar texto plano y a la vez mostrar la información devuelta por este. Parecido a cómo trabajan los clientes de Telnet.

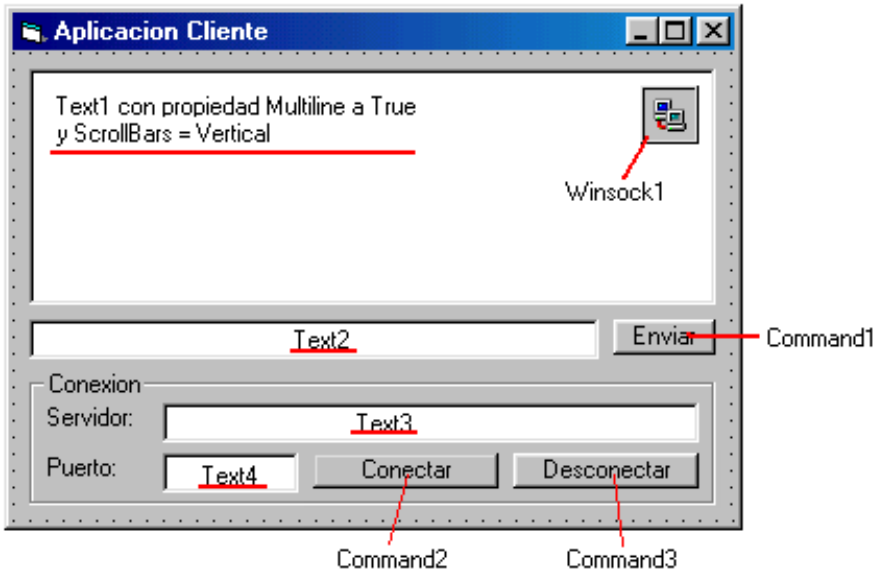


Figura 0.24 Formulario de interfaz de usuario para el cliente.

La primera acción a realizar y fundamental para toda aplicación de este tipo, es crear la conexión al servidor, ya que solo se puede transmitir información si la conexión cliente/servidor se encuentra activa.

Propiedades necesarias

- RemoteHost: Asignamos la dirección a la que deseamos conectar.

- RemotePort: Asignamos el puerto al que deseamos conectar en RemoteHost.

Métodos necesarios

- Connect(): Conecta al servidor.

- Close(): Cierra la conexión al servidor.

Eventos involucrados

- Connect(): Ocurre cuando hemos establecido con éxito la conexión al servidor

- Close(): Ocurre cuando el servidor nos cierra la conexión.

- Error(): Ocurre en caso de errores.

Para realizar la conexión utilizamos el siguiente código:

```
Private Sub Command2_Click()            'asignamos los datos de conexión
Winsock1.RemoteHost = Text3.Text
Winsock1.RemotePort = Text4.Text
Winsock1.Close
Winsock1.Connect                        'conectamos el socket
End Sub
```

Aquí se pueden ver claramente dos partes principales:

En las primeras dos líneas se asignan los datos de conexión al host remoto, como son la IP/DNS (RemoteHost) y Puerto (RemotePort).

En la última línea se llama al método "Connect" para realizar la conexión, siempre asegurándonos que el Socket no esté utilizándose. Para ello llamamos al método "Close" que se encarga de cerrar toda conexión pendiente en el Socket.

Nota: También se puede especificar los datos de conexión (IP y Puerto) directamente en el comando "Connect" como parámetros, de la sig. Forma: Winsock1.Connect(Host, Puerto).

Si la conexión se realiza con éxito se dispara un evento para tal fin, en donde podemos realizar acciones inmediatas en el momento preciso en que se logra establecer la conexión con el servidor. El evento es el siguiente:

```
Private Sub Winsock1_Connect()          'desplegamos un mensaje en la ventana
Text1.Text = Text1.Text & "*** Conexión establecida." & vbCrLf 'desplazamos el scroll
Text1.SelStart = Len(Text1.Text)
End Sub
```

En este caso solo se limita a mostrar un mensaje en pantalla especificando que la conexión se ha realizado con éxito.

También se debe tener presente que en cualquier momento el servidor puede cerrar la conexión, o bien cerrarse por algún error, para ello es que se cuenta con el evento "Close", que se dispara al perder la conexión con el servidor:

```
Private Sub Winsock1_Close()            'cierra la conexión
Winsock1.Close
Text1.SelStart = Len(Text1.Text)        'desplegamos un mensaje en la ventana
Text1.Text = Text1.Text & "*** Conexion cerrada por el servidor." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

Aquí solo se despliega un mensaje en la pantalla informando del evento ocurrido, y cerrando previamente el Socket para asegurarnos de que este actualice sus valores según el estado actual.

En cambio si se desea cerrar la conexión con el servidor de manera manual, basta con llamar al método "Close" directamente:

```
Private Sub Command3_Click()                'cierra la conexión

Winsock1.Close
Text1.Text = Text1.Text & "*** Conexión cerrada por el usuario." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

Una vez realizada con éxito la conexión, resta comenzar a transferir datos, cabe mencionar que estos datos se envían siempre en forma binaria.

Métodos necesarios

- **SendData**: Envía datos al otro extremo de la conexión (socket remoto).

- **GetData**: Recibe datos enviados por el extremo remoto (socket remoto).

Eventos involucrados

- **DataArrival():** Ocurre cuando el socket remoto nos está enviando datos.

- **Error():** Ocurre en caso de errores.

Para enviar datos utilizamos el método "SendData" de la siguiente forma:

```
Private Sub Command1_Click()
Winsock1.SendData Text2.Text & vbCrLf
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
Text1.Text = Text1.Text & "Cliente >" & Text2.Text & vbCrLf 'mostramos los datos
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
'borramos Text2
Text2.Text = ""
End Sub
```

Al método SendData solo se le pasa como parámetro el dato a enviar (en este caso el contenido de un TextBox más los caracteres de nueva línea y retorno de carro) y este lo envía inmediatamente al socket remoto.

Cuando el socket remoto nos envía un dato (de la misma forma que realizamos anteriormente) se genera el evento "DataArrival()" indicando que tenemos nueva información disponible, y esta información la recogemos con el método "GetData":

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Buffer As String 'variable para guardar los datos
Winsock1.GetData Buffer
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
Text1.Text = Text1.Text & "Servidor >" & Buffer 'mostramos los datos
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
End Sub
```

En este ejemplo solo se obtiene los datos y lo mostramos inmediatamente en la ventana del cliente, no se hace ningún tratamiento previo de los datos, como sería lo habitual.

2.5.8.6.APLICACIÓN SERVIDOR.

Se va a realizar una aplicación que se mantenga a la escucha de una conexión entrante y la acepte, podrá enviar y recibir datos desde el cliente.

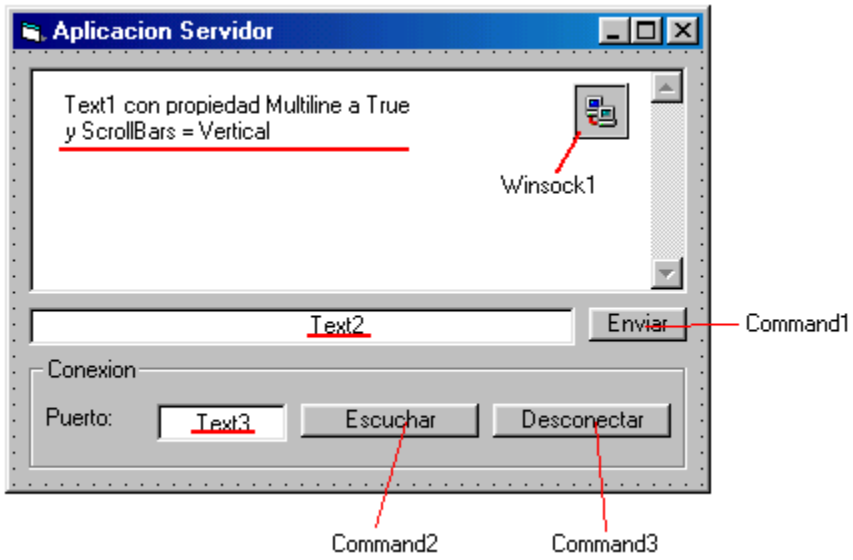


Figura 0.25 Formulario de interfaz de usuario para el servidor.

Al igual que en el cliente, lo primero es habilitar el socket para que pueda quedar esperando una conexión, se dice que queda "a la escucha de". Para esto solo se necesita un botón "Escuchar" y como datos un puerto local (a elección) en el cual deseamos recibir conexiones entrantes.

Propiedades necesarias

- LocalPort: Asigna el puerto local en el cual deseamos recibir conexiones.

Métodos necesarios

- Listen(): Escucha peticiones entrantes.

- Close(): Cierra la conexión al servidor.

Eventos involucrados

- ConnectionRequest(): Ocurre cuando un cliente nos solicita una conexión al servidor.

- Close(): Ocurre cuando el servidor nos cierra la conexión.

- Error(): Ocurre en caso de errores.

El código utilizado para el botón "Escuchar" es el siguiente:

```
Private Sub Command2_Click()
'cerramos cualquier conexión previa
Winsock1.Close
'asignamos el puerto local que abriremos
Winsock1.LocalPort = Text3.Text
'deja el socket escuchando conexiones
Winsock1.Listen
'desplegamos un mensaje en la ventana
Text1.SelStart = Len(Text1.Text)
Text1.Text = Text1.Text & "*** Escuchando conexiones." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

La primera línea de código cierra la conexión actual, para luego poder modificar los datos y crear una nueva conexión sin que de errores.

La siguiente línea le dice en que puerto deseamos recibir conexiones, y luego llama al socket para que quede a la escucha de conexiones en ese puerto.

Hasta aquí el socket solo está "escuchando" conexiones, es decir aun nadie se puede conectar al servidor completamente.

Esto solo nos permite avisarnos cada vez que un cliente se quiera conectar o bien cada vez que un cliente "Solicita una conexión entrante". Cuando este sucede se genera el evento *"ConnectionRequest()":*

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
'mostramos un mensaje en la ventana
Text1.SelStart = Len(Text1.Text)
Text1.Text = Text1.Text & "*** Petición número " & requestID & vbCrLf
Text1.SelStart = Len(Text1.Text)
'cerramos previamente el socket
Winsock1.Close
'aceptamos la conexión
Winsock1.Accept requestID
'desplegamos un mensaje en la ventana
Text1.SelStart = Len(Text1.Text)
Text1.Text = Text1.Text & "*** Conexión aceptada, listo para interactuar." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

Lo primero que se había realizado es dejar el socket a la escucha de conexiones (para esto se utiliza el método "Listen"). Con esto ya tenemos un puerto abierto y atento a toda actividad.

Cuando un "Cliente" se intenta conectar a ese puerto, el socket lo detectara y para ello generara el evento "ConnectionRequest()" que significa "Petición de conexión" y además le asigna una identidad a esa "Petición" que identifica al "Cliente" remoto. Esta identidad es pasada como parámetro en el evento "ConnectionRequest()" con el nombre de "requestID" y es de tipo "Long".

Cuando se genera el evento lo que tenemos que hacer es "Aceptar" la conexión entrante "requestID" mediante el método "Accept", si no lo hacemos al llegar al "End Sub" del evento, la conexión del "Cliente" será cerrada automáticamente.

Es de ver que antes de aceptar la conexión con "Accept" primero se cierra la conexión con "Close", esto que puede parecer ilógico no lo es, porque el socket lo teníamos ocupado y activo "escuchando conexiones", y ahora necesitamos que establezca una conexión par a par con el cliente, por ello es que cerramos la función de "Escuchar conexiones del socket" y le decimos que acepte la conexión entrante y así automáticamente se conecta en forma directa con el cliente y ya no entenderá nuevas conexiones entrantes. (No puede realizar dos funciones a la vez)

Para cerrar la conexión basta con usar el método "Close" en cualquier momento:

```
Private Sub Command3_Click()
'cierra la conexión
Winsock1.Close
'desplegamos un mensaje en la ventana
Text1.SelStart = Len(Text1.Text)
Text1.Text = Text1.Text & "*** Conexión cerrada por el usuario." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub

Private Sub Winsock1_Close()
'cierra la conexión
Winsock1.Close
'desplegamos un mensaje en la ventana
Text1.SelStart = Len(Text1.Text)
Text1.Text = Text1.Text & "*** Conexión cerrada por el Cliente." & vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

Envío y recepción de datos.
Esto es idéntico al explicado en la parte del cliente:

```
Private Sub Command1_Click()
'enviamos el contenido de Text2
Winsock1.SendData Text2.Text & vbCrLf
'apuntamos al final del contenido del TextBox e
'insertamos los nuevos datos obtenidos
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
Text1.Text = Text1.Text & "Servidor >" & Text2.Text & vbCrLf 'mostramos los datos
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
'borramos Text2
Text2.Text = ""
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Buffer As String 'variable para guardar los datos
'obtenemos los datos y los guardamos en una variable
Winsock1.GetData Buffer
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
Text1.Text = Text1.Text & "Cliente >" & Buffer 'mostramos los datos
Text1.SelStart = Len(Text1.Text) 'coloca el cursor al final del contenido
End Sub
```

Manejo de errores.

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal Scode As
Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
'cerramos la conexion
Winsock1.Close
'mostramos informacion sobre el error
MsgBox "Error número " & Number & ": " & Description, vbCritical
End Sub
```

2.6. SQL.

El lenguaje de consulta estructurado (SQL) es un lenguaje de base de datos normalizado, utilizado por el motor de base de datos de Microsoft Jet. SQL se utiliza para crear objetos QueryDef, como el argumento de origen del método OpenRecordSet y como la propiedad RecordSource del control de datos. También se puede utilizar con el método Execute para crear y manipular directamente las bases de datos Jet y crear consultas SQL de paso a través para manipular bases de datos remotas cliente - servidor.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

2.6.1. COMANDOS.

Existen dos tipos de comandos SQL:

- Los DLL que permiten crear y definir nuevas bases de datos, campos e índices.
- Los DML que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Comandos DLL.

CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DML.

SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

2.6.2. CLAUSULAS.

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

2.6.3. OPERADORES LÓGICOS.

AND	Es el "y" lógico. Evaluá dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.

OR Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT Negación lógica. Devuelve el valor contrario de la expresión.

2.6.4. OPERADORES DE COMPARACIÓN.

< Menor que
> Mayor que
<> Distinto de
<= Menor o Igual que
>= Mayor o Igual que
= Igual que
BETWEEN Utilizado para especificar un intervalo de valores.
LIKE Utilizado en la comparación de un modelo
In Utilizado para especificar registros de una base de datos

2.6.5. FUNCIONES DE ARREGLO.

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

AVG Utilizada para calcular el promedio de los valores de un campo determinado
COUNT Utilizada para devolver el número de registros de la selección
SUM Utilizada para devolver la suma de todos los valores de un campo determinado
MAX Utilizada para devolver el valor más alto de un campo especificado
MIN Utilizada para devolver el valor más bajo de un campo especificado

2.6.6. CONSULTAS DE SELECCIÓN.

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros que se pueden almacenar en un objeto recordset. Este conjunto de registros es modificable.

2.6.6.1.CONSULTAS BÁSICAS.

La sintaxis básica de una consulta de selección es la siguiente:

```
SELECT Campos FROM Tabla;
```

En donde campos es la lista de campos que se deseen recuperar y tabla es el origen de los mismos, por ejemplo:

```
SELECT Nombre, Telefono FROM Clientes;
```

2.6.7. CONSULTAS DE ACCIÓN.

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir y borrar y modificar registros.

2.6.7.1.DELETE.

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

```
DELETE Tabla.* FROM Tabla WHERE criterio
```

DELETE es especialmente útil cuando se desea eliminar varios registros. En una instrucción DELETE con múltiples tablas, debe incluir el nombre de tabla (Tabla.*). Si especifica más de una tabla desde la que eliminar registros, todas deben ser tablas de muchos a uno. Si desea eliminar todos los registros de una tabla, eliminar la propia tabla es más eficiente que ejecutar una consulta de borrado.

Se puede utilizar DELETE para eliminar registros de una única tabla o desde varios lados de una relación uno a muchos. Las operaciones de eliminación en cascada en una consulta únicamente eliminan desde varios lados de una relación. Por ejemplo, en la relación entre las tablas Clientes y Pedidos, la tabla Pedidos es la parte de muchos por lo que las operaciones en cascada solo afectaran a la tabla Pedidos. Una consulta de borrado elimina los registros completos, no únicamente los datos en campos específicos. Si desea eliminar valores en un campo especificado, crear una consulta de actualización que cambie los valores a Null.

Una vez que se han eliminado los registros utilizando una consulta de borrado, no puede deshacer la operación. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad.

```
DELETE * FROM Empleados WHERE Cargo = 'Vendedor';
```

2.6.7.2.INSERT INTO.

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: Insertar un único registro o Insertar en una tabla los registros contenidos en otra tabla.

En este caso la sintaxis es la siguiente:

```
INSERT INTO Tabla (campo1, campo2, .., campoN) VALUES (valor1, valor2, ..., valorN);
```

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres) y las fechas indicarlas en formato mm-dd-aa.

```
INSERT INTO Clientes SELECT Clientes_Viejos.* FROM Clientes_Nuevos;
INSERT INTO Empleados (Nombre, Apellido, Cargo)VALUES ('Luis', 'Sánchez');
```

2.6.7.3.UPDATE.

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

```
UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2, ... CampoN=ValorN
WHERE Criterio;
```

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez. El ejemplo siguiente incrementa los valores Cantidad pedidos en un 10 por ciento y los valores Transporte en un 3 por ciento para aquellos que se hayan enviado al Reino Unido.:

```
UPDATE Pedidos SET Pedido = Pedidos * 1.1, Transporte = Transporte * 1.03
WHERE PaisEnvío = 'ES';
```

UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

```
UPDATE Empleados SET Grado = 5 WHERE Grado = 2;
UPDATE Productos SET Precio = Precio  * 1.1 WHERE Proveedor = 8 AND Familia = 3;
```

Si en una consulta de actualización suprimimos la cláusula WHERE todos los registros de la tabla señalada serán actualizados.

UPDATE Empleados SET Salario = Salario * 1.1

2.7. BASE DE DATOS ACCESS.

Una base de datos se define como un conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, con la posibilidad de efectuar consultas, ordenaciones en base a diferentes criterios, imprimir informes, etc.

El objetivo principal de una base de datos es unificar toda la información del sistema para evitar redundancias, sin perder las distintas perspectivas, que de la misma tienen los usuarios, Los datos se organizan y se mantienen en un conjunto estructurado que no está diseñado para una aplicación concreta, sino que tiende a satisfacer las necesidades de información de toda la organización.

Las herramientas software desarrollado para soportar el concepto de base de datos se conocen como Sistemas Gestores de Bases de Datos (SGBD). Un SGBD se caracteriza por:

- Descripción centralizada de todos los datos.
- Posibilidad de definir vistas parciales de dichos datos para los distintos usuarios.

2.7.1. ACCESS COMO GESTOR DE BASE DE DATOS.

Access es un programa que permite definir, desarrollar, mantener y utilizar bases de datos relacionales. Este tipo de herramientas se llama habitualmente generadores de aplicaciones.

Access permite integrar información gráfica en forma de imágenes y dibujos con texto e información numérica. En definitiva, es una herramienta de base de datos que permite crear aplicaciones de una forma rápida y sencilla.

Las bases de datos relacionales facilitan la búsqueda, el análisis, el mantenimiento y la protección de los datos almacenados, pues los datos se almacenan en un solo lugar.

2.7.2. CREAR UNA BASE DE DATOS ACCESS 2010.

Para crear una nueva base de datos debemos: Hacer clic sobre la opción **Nuevo** de la pestaña **Archivo**.

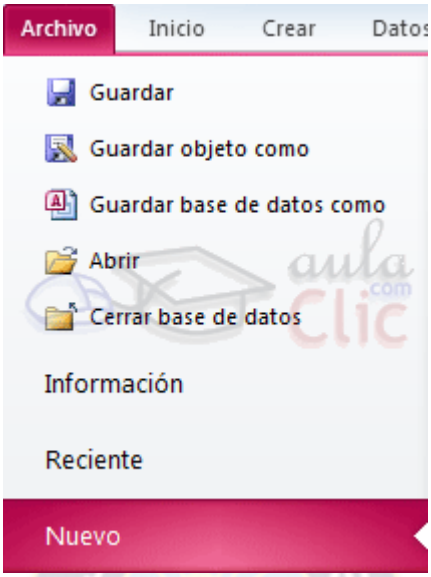


Figura 0.26 Menú de opciones de access 2010.

Se mostrarán las distintas opciones para nuevos archivos en la zona de la derecha. La opción **Base de datos en blanco** es la que debe estar seleccionada si queremos partir de cero, aunque también podríamos partir de una base existente o de una plantilla.

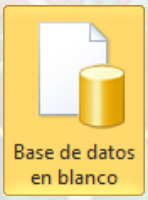


Figura 0.27 Icono de base de datos en blanco.

U poco más a la derecha verás un panel con una vista previa y las opciones necesarias para especificar el **Nombre de archivo** y **Ubicación** de la nueva base de datos.

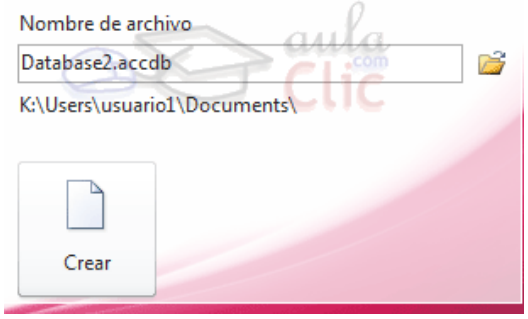


Figura 0.28 Introducción de la ruta para crear la base de datos.

Para **cambiar la ubicación** establecida por defecto (la carpeta ***Mis documentos***), se hace clic en la pequeña carpeta que hay junto a la caja de texto. Se abrirá el siguiente cuadro de diálogo:

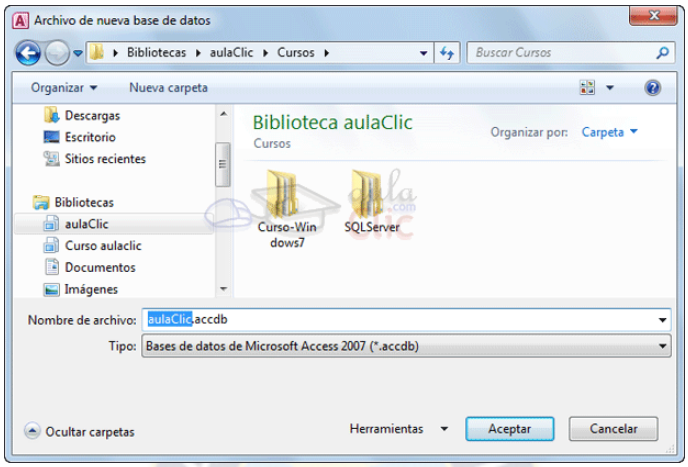


Figura 0.29 Selección de nueva ruta para la creación de la base de datos

Luego, hacer clic en **Aceptar**. Se cerrará el cuadro de diálogo y volverás a la pantalla anterior.

Pulsar el botón **Crear** para crear la base de datos.

Automáticamente se creará nuestra nueva base de datos a la cual Access asignará la extensión **.ACCDB**. Es el mismo formato que se utilizaba en Office 2007, por lo que no presentan problemas de compatibilidad.

Por defecto, Access abrirá una nueva **tabla** llamada ***Tabla1*** para que puedas empezar a rellenar sus datos.

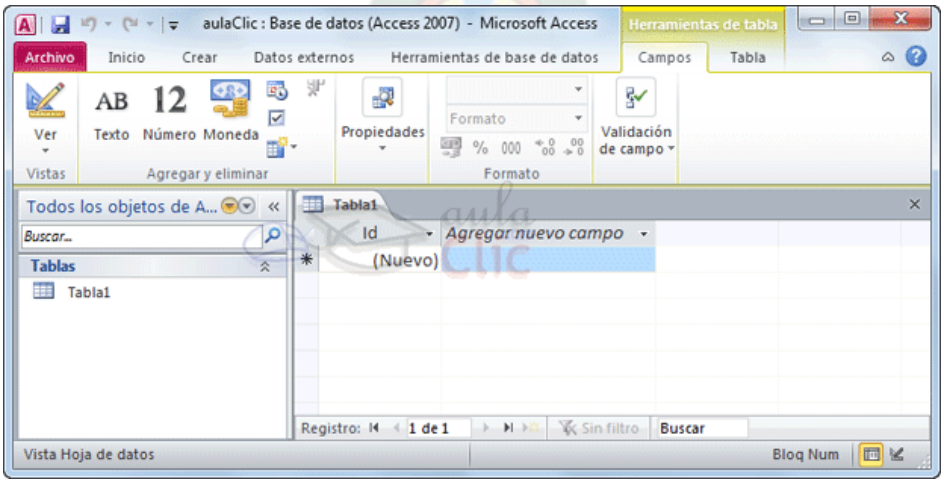


Figura 0.30 Base de datos creada.

Una tabla es el elemento principal de cualquier base de datos ya que todos los demás objetos se crean a partir de éstas.

Si observas esta ventana, a la izquierda aparece el **Panel de navegación**, desde donde podremos seleccionar todos los objetos que sean creados dentro de la base de datos. De entrada sólo encontraremos la ***Tabla1*** que se crea por defecto.

2.8. MODEM GSM – GPRS ENFORA GSM 1218/GSM 1308.

El Sistema Global para las comunicaciones móviles (GSM) es un sistema estándar de comunicación inalámbrica. Por medio de esta red es posible el intercambio de información, principalmente de equipos móviles. Con ello es posible enviar o recibir tanto Voz, Datos y mensajería SMS.

2.8.1. COMANDOS AT.

Los comandos AT son instrucciones codificadas que conforman el lenguaje de comunicación entre un usuario y un terminal módem y son de carácter genérico en su mayoría, ya que un mismo comando funciona en modelos de distintas marcas, haciendo que un programa basado en comandos AT sea inmensamente robusto y compatible con la mayor parte de los dispositivos disponibles en el mercado. La gran parte de los módems disponibles reconocen los comandos AT más utilizados. Por lo mismo, la tecnología GSM ha adaptado el uso de estos comandos, teniendo comandos específicos que pueden ser encontrados en documentación especializada sobre el módulo GSM. Dependiendo del módulo usado, es la implementación que se le da a los comandos y no depende del medio de comunicación, que puede ser serial, infrarrojo o Bluetooth

Los comandos AT, poseen en su mayoría un prefijo dado por “AT”. Cada acción que se desee viene precedida por este prefijo. Así por ejemplo, si se quiere obtener información de identificación del fabricante, se debe ingresar el comando AT+CGMI, donde en este caso se obtendrá como respuesta Enfora, Inc. Si se desea información sobre la identificación del modelo se debe ingresar AT+CGMM, obteniendo como respuesta Enabler-II G Módem. Los comandos se pueden ingresar tanto con minúsculas como con mayúsculas.

2.8.2. COMANDOS AT BÁSICOS.

Existen comandos generales bastante útiles los cuales permiten, entre otras cosas, evitar escribir una y otra vez los mismos comandos al reiniciar el módulo. Estos comandos permiten entre otras

cosas, guardar, borrar y mostrar la configuración dada al módem. No todos los parámetros se ven afectados por estos comandos, pero los más usados sí. Estos comandos son:

ATZ. Este comando, ajusta los parámetros a su valor por defecto, es decir, ajusta los parámetros a los valores que tenían al momento en que se encendió el módem. Entrega por respuesta un OK

AT&F. Ajusta los parámetros a los valores dados de fábrica. Estos valores son en general, desactivación total de cualquier aviso, además de desactivar la conexión automática a la red GPRS.

AT&W. Para guardar los datos configurados hasta ahora, se puede utilizar el comando AT&W, el cual los guarda en la memoria y aunque se reinicie la tarjeta, estos valores continúan.

AT&V. Este comando entrega una lista de los valores de los parámetros que actualmente posee el módem.

2.8.3. COMANDOS DE LLAMADA.

Para realizar la llamada se utiliza el siguiente comando:

ATD<número>

EL formado del número puede contener dígitos de discado (0-9,*,#,+,A,B,C) y dígitos modificadores (, (comma), T, P, !,@, W). Un ejemplo de discado sería:

ATD1234567890; disca el número 1234567890, estando en el modo DATA (at+fclass=0).

Como respuesta a la llamada de voz se puede obtener:

NO DIAL TONE Si es que no se detecta un tono de discado.

NO CARRIER Si la llamada no se puede realizar. Aparece también cuando, una vez establecida la conexión, el otro lado de la línea cuelga la llamada. Se debe revisar la potencia de la señal, la banda y si la tarjeta se encuentra registrada por el proveedor de red por medio de at+csq?, at%band? y at+creg? respectivamente.

CONNECT <value> Cuando se conecta en el modo DATA (at+fclass=8), el valor de <value> dependerá de la configuración del comando ATX. Puede mostrar el valor de IP donde se conecta.

BUSY Si es que el número discado se encuentra ocupado.

OK Cuando la llamada fue exitosa y retorna al modo de comandos.

Cuando llega un mensaje **RING**, se puede o contestar la llamada o rechazarla.

ATA. Contesta una llamada entrante

ATH o AT+CHUP, PARA COLGAR SE UTILIZA EL COMANDO:

ATH Cuelga una llamada que se está realizando o ya contestaron.

AT+CHUP Cuelga todas las llamadas actuales conectadas.

La principal diferencia entre ambas, es que si no se ha contestado una llamada que se está realizando, se debe ocupar ATH para colgar y no AT+CHUP, ya que este último comando no realiza ninguna acción que no sea colgar una llamada ya conectada o contestada.

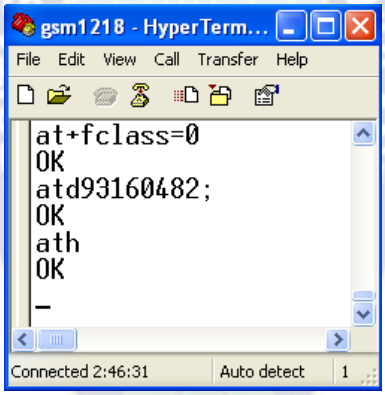


Figura 0.31 Ejemplo de una llamada telefónica.

2.8.4. COMANDOS PARA ENVÍOS DE MENSAJES DE TEXTO.

AT+CMGF

AT+CMGF=<mode> Configura el tipo de formato de los mensajes de texto SMS.

Donde:

<mode> 0 formato PDU para mensajes SMS.

1 formato TEXTO para mensajes SMS.

AT+CMGS= “<numero>”

> <Mensaje de texto>, <CTRL+Z>

Este comando crea un mensaje de texto y lo envía inmediatamente al destinatario dado por <numero> (debe ser escrito entre comillas). Se debe escribir el comando, luego el número y presionar ENTER y así esperar hasta que aparezca el carácter '>'.

Posteriormente se escribe el mensaje te texto deseado y para terminar se presionan las teclas CTRL+Z, lo cual procederá a enviar el mensaje. Como respuesta se obtiene:

+CMGS: <mr>

Donde <mr> es un índice de referencia del mensaje enviado.

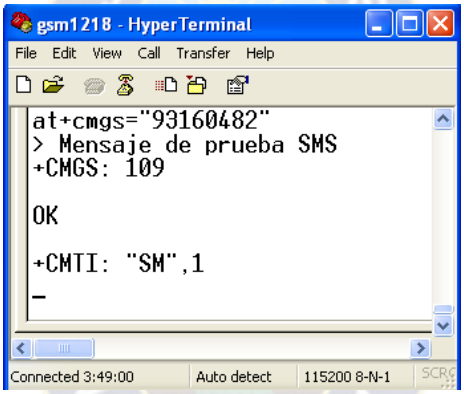


Figura 0.32 Ejemplo de envío de mensaje.

2.8.5. COMANDO PARA DETERMINAR EL NIVEL SE SEÑAL.

AT+CSQ

AT+CSQ Indica el nivel potencia de la señal y la calidad de ésta. Su formato de salida es:

+CSQ: <rssi>,<ber>
Dónde:
<rssi> Indicador de la potencia de la señal recibida.
0 -113 dBm o menos
1 -111 dBm
2-30 -109 -53 dBm.
31 -51 dBm o más.
99 desconocido o no detectable.
<ber> Indica en porcentaje el bit error rate del canal.
0-7 entre 0.1% y 15% o más.
99 desconocido o no detectable

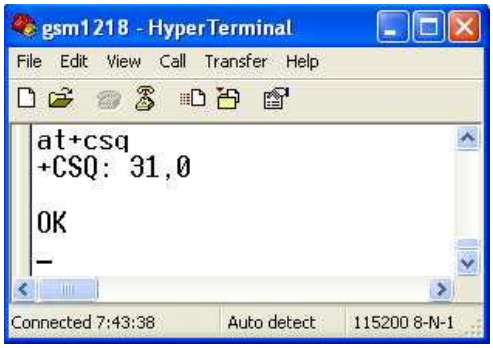


Figura 0.33 Ejemplo de cónsula sobre el nivel se señal.

2.8.6. COMANDOS PARA LA CONEXIÓN A UN SERVIDOR.

Los comandos requeridos para efectuar la conexión a un servidor son los siguientes.

AT+CREG=1	, se registra a la red GSM.
AT+CGREG=1	, se registra a la reg GPRS.
at+cgdcont=1,"ip","APN","",0,0	, se introduce el APN del operador.
AT$ACTIVE=1	, se conecta como cliente.
AT$HOSTIF=2	, llamada de datos para utilizar TCP PAD.
AT$PADSRC=n	, número de puerto del cliente.
AT$PADDST="ip",nn	, ip y puerto del servidor.
ATX1	, muestra el ip asignado al cliente.
AT+CGPADDR	, entrega el ip asignado al modem.
ATD*99***1#	, inicia la conexión con el servidor.

2.9. MICROCONTROLADOR PIC 18f4550.

El PIC 18f4550 es un microcontrolador fabricado por MICROCHIP, las características más relevantes son.

2.9.1. CARACTERÍSTICAS DE POTENCIA.

Modo RUN, CPU encendida y periféricos encendidos.

Modo IDLE, CPU apagado, periféricos encendidos.

Modo SLEEP, CPU apagado, periféricos apagados.

En modo IDLE, el consumo de corriente es 5.8 uA típico.

En modo SLEEP, el consumo de corriente es 0.1uA típico.

El oscilador del temporizador 1 alcanza a consumir 1.1uA a 2V y 32KHz.

El temporizador del perro guardián consume 2.1uA típico a 2v y 32KHz.

2.9.2. CARACTERÍSTICAS DE LOS PERIFÉRICOS.

Cuenta con tres interrupciones externas.

Cuatro módulos temporizadores.

Dos módulos (CCP) Captura, Comparación y Pwm.

- Captura de 16 bits, resolución máxima de 5.2ns.
- Comparación de 16 bits, resolución máxima de 83.3ns.
- Pwm con resolución de 10 bits.

Captura / Comparación / PWM (PECC) Módulo mejorado:

- Múltiples modos de salida.
- Polaridad seleccionable.
- Programable tiempo muerto.
- Auto-apagado y reinicio automático.

Módulo USART mejorado:

- Soporte de bus LIN.

Módulo maestro Synchronous Serial Port (MSSP), apoyando 3-Wire SPI (los 4 modos) y I2C en modos maestro y esclavo.

Trece canales de conversor analógico a digital de 10 Bits de resolución con módulo programable Tiempo de adquisición.

Dos comparadores analógicos con multiplexación de entrada.

2.9.3. UNIVERSAL SERIAL BUS CARACTERÍSTICAS.

USB V2.0

Baja velocidad (1,5 Mb / s) y Full Speed (12 Mb / s)

Soporta control de interrupción, isócrona y Granel

Soporta hasta 32 puntos finales (16 bidireccionales)

1 Kbyte RAM de acceso dual para USB

On Chip Transceptor USB con On-Chip Voltaje regulador

Interfaz para Off-Chip Transceptor USB

Streaming Parallel Port (SPP) para la transmisión de USB transferencias (dispositivos 40/44-pin solamente).

2.9.4. CARACTERÍSTICAS ESPECIALES DEL MICROCONTROLADOR.

Arquitectura Optimizada para compiladores C con opcional set de instrucciones extendida.

Memoria Flash con 100.000 Ciclo de borrado / escritura.

Memoria EEprom típico 1.000.000 Ciclos borrado y escritura de datos.

Flash / EEPROM de datos de retención > 40 años.

Auto-programable bajo control de software.

Niveles de prioridad de interrupciones.

8 x 8 Solo-ciclo Multiplicador Hardware.

Watchdog Timer (WDT):

- Periodo Programable de 41 ms a 131s

Protección de código programable.

Alimentación de 5V.

Amplio rango de tensión de funcionamiento (2.0V a 5.5V).

2.9.5. ARQUITECTURA DE PIC 18f4550.

Arquitectura RISC avanzada Harvard: 16 bit con 8 bit de datos.

CARACTERISTICAS	PIC18F4450
Frecuencia de Operación	Hasta 48MHz
Memoria de Programa (bytes)	32.768
Memoria RAM de Datos (bytes)	2.048
Memoria EEPROM Datos (bytes)	256
Interrupciones	20
Líneas de E/S	35
Temporizadores	4
Módulos de Comparación/Captura/PWM (CCP)	1
Módulos de Comparación/Captura/PWM mejorado (ECCP)	1
Canales de Comunicación Serie	MSSP,EUSART
Canal USB	1
Puerto Paralelo de Transmisión de Datos (SPP)	1
Canales de Conversión A/D de 10 bits	13 Canales
Comparadores analógicos	2
Juego de instrucciones	75 (83 ext.)
Encapsulados	PDIP 40 pines QFN 40 pines TQFP 40 pines

Figura 0.34 Cuadro de características del PIC18f4550.

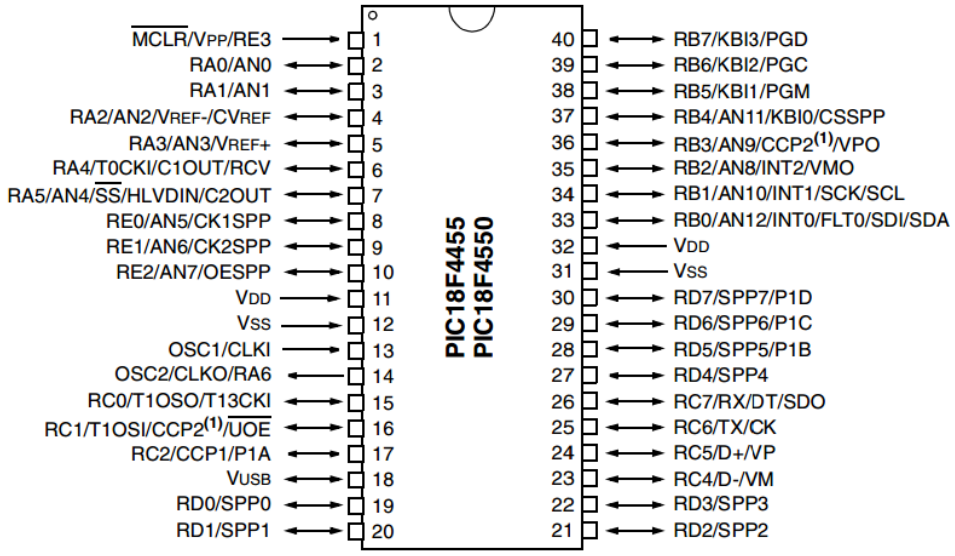


Figura 0.35 Disposición de pines del PIC18f4550.

Diagrama de bloques.

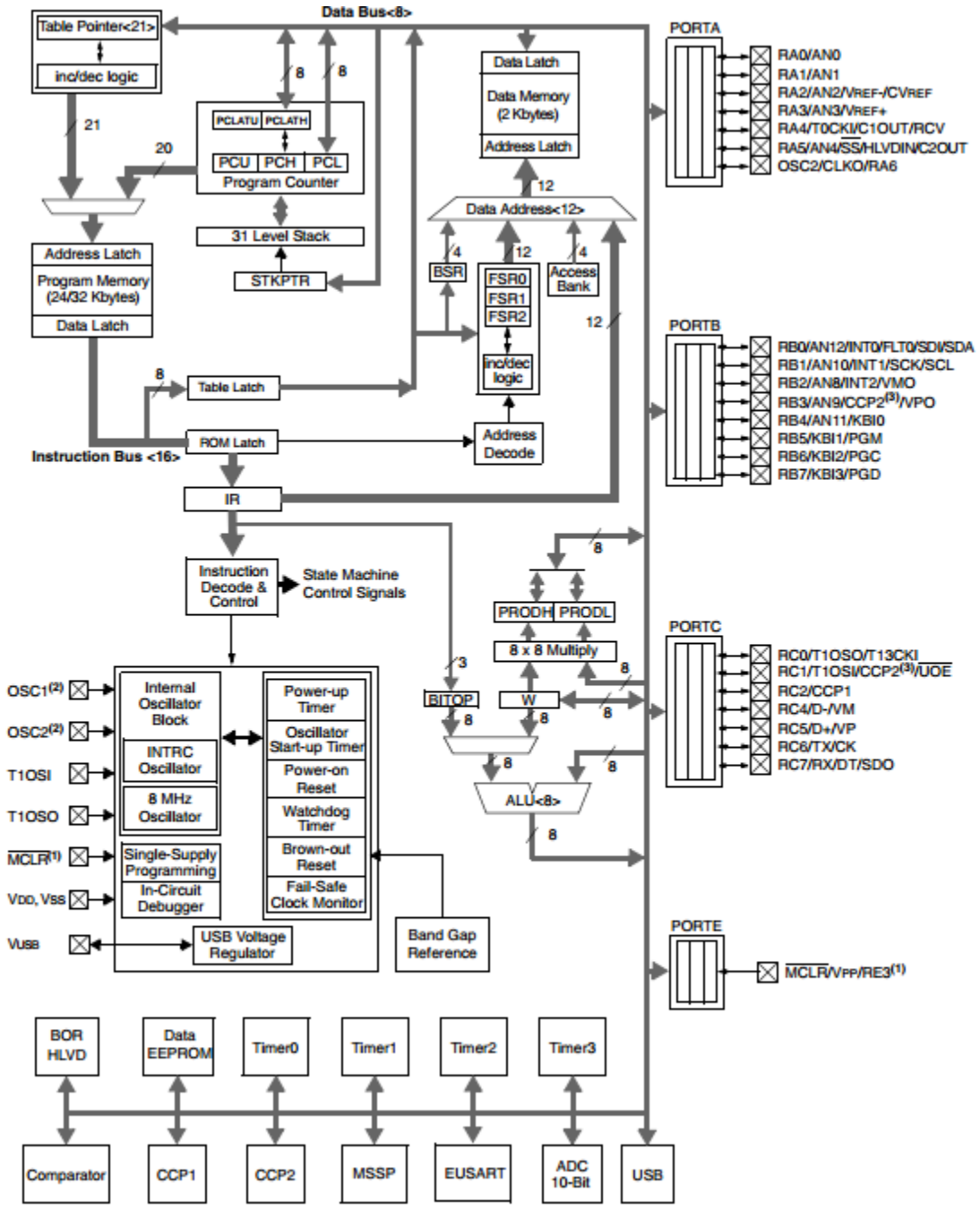


Figura 0.36 Diagrama de bloques del PIC18f4550.

2.9.5.1.MEMORIA.

El Pic 18F4550 dispone de diversos tipos de memoria:

Memoria de programa: memoria flash interna de 32.768 bytes, la cual almacena las instrucciones y constantes/datos.

Puede ser escrita o leída mediante un programador externo o durante la ejecución del programa.

Memoria RAM de datos: Memoria SRAM interna de 2048 bytes en los que se incluyen los registros de función especial.

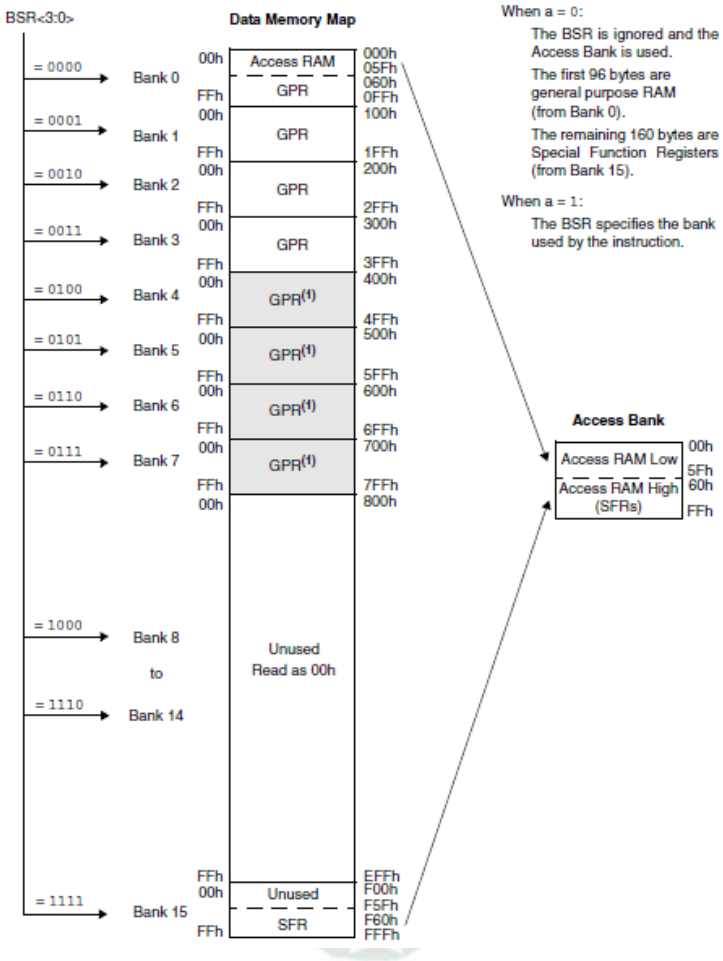


Figura 0.37 Mapa de memoria RAM del PIC18f4550.

Almacena datos que de forma temporal durante la ejecución del programa.

Se puede leer/escribir durante su ejecución mediante varias instrucciones.

Para acceder a un byte de la memoria RAM de datos, primero se selecciona el banco al que pertenece el byte mediante el registro de selección de banco (BSR) y después direccionar el byte dentro del banco.

Los bancos 4, 5, 6 y 7 se utilizan también como buffer para el puerto USB.

Memoria EEPROM de datos: memoria no volátil de 256 bytes.

Almacena datos que se conservan aunque el sistema carezca de alimentación.

Se puede leer/escribir durante su ejecución por medio de registros.

Pila: bloque de 31 palabras de 21 bits.

Almacena la dirección de instrucción que se debe ejecutar después de una interrupción o subrutina.

Memoria de configuración: memoria en la que se incluyen los bits de configuración (12 bytes de memoria flash) los registros de identificación (2 bytes de memoria solo lectura.)

Se sitúa más allá de la zona de memoria de programa usuario, y en esta configuración se incluyen:

Bits de configuración contenidos en 12 bytes de memoria flash que permiten la configuración de algunas opciones de microcontrolador tales como:

Opciones del oscilador.
Opciones de reset.
Opciones del watchdog.
Opciones de la circuitería de depuración y programación.
Opciones de protección contra escritura de memoria de programa y memoria

SFR. La memoria RAM de datos se compone de registros de propósito general (GPR´s) y de los registros de función especial (SFR´s).

Mediante estos últimos se puede controlar el funcionamiento de la CPU y de las unidades funcionales del microcontrolador.

Reloj.

En este microcontrolador el oscilador puede ser externo al igual que el resto de microcontroladores pero también puede ser interno ya que cuenta el oscilador que interno tiene la posibilidad de funcionar desde los 31 KHz hasta su máxima velocidad que es 8 MHz.

2.10. COMPILADOR CCS PICC.

El compilador C de del fabricante CCS ha sido desarrollado específicamente para el PIC, Dispone de una amplia librería de funciones predefinidas, comandos de procesado y ejemplos, adicionalmente proporciona los controladores para diversos dispositivos como LCD, conversores analógicos a digitales, relojes en tiempo real, eeproms seriales, etc.

2.10.1. ESTRUCTURA DE UN PROGRAMA EN CCS PICC.

Para escribir un programa en C mediante el compilador CCS se deben temar en cuenta una serie de elementos básicos de su estructura.

- **DIRECTIVAS DE PROCESADO.**

Controlan la conversión del programa a código maquina por parte del compilador.

- **PROGRAMAS O FUNCIONES.**

Es el conjunto de instrucciones. Puede haber uno o varios, en cualquier caso siempre debe haber uno definido como principal mediante la instrucción llamada *main().*

- **INSTRUCCIONES.**

Indican cómo se debe comportar el PIC en todo momento.

- **COMENTARIOS.**

Permiten describir lo que significa cada línea del programa.

Ejemplo.

```
#include <18f4550.h>          //*********************************
#fuses   hs,nowdt             //directivas
#use     delay(clock=4000000) //*********************************
void main()                   //*********************************
{                             //procedimiento principal
  while(true)                 //
  {                           //
    output_toggle(pin_a0);    //
    delay_ms(200);            //
  }                           //
}                             //*********************************
```

2.10.2. TIPOS DE VARIABLES.

Tipo	Tamaño	Rango	Descripción
Int1 Short	1 bit	0 a 1	Entero de 1 bit
Tipo	**Tamaño**	**Rango**	**Descripción**
Int Int8	8 bit	0 a 255	Entero
Int16 Long	16 bit	0 a 65.535	Entero de 16 bit
Int32	32 bit	0 a 4.294.967.295	Entero de 32 bit
Float	32 bit	$\pm 1.175 \times 10^{-38}$ a $\pm 3.402 \times 10^{+38}$	Coma flotante
Char	8 bit	0 a 255	Carácter
Void	-	-	Sin valor
Signed Int8	8 bit	-128 a +127	Entero con signo
Signed Int16	16 bit	-32768 a + 32767	Entero largo con signo
Signed Int32	32 bit	-2^{31} a $+(2^{31}-1)$	Entero 32 bit con signo

2.10.3. LAS CONSTANTES.

Las constantes pueden ser especificadas en decimal, octal, hexadecimal o binario

123	Decimal
0123	Octal (**0**)
0x123	Hexadecimal(**0x**)
0b010010	Binario (**0b**)
'x'	Carácter
'\010'	Carácter octal
'\xA5'	Carácter hexadecimal

2.10.4. VARIABLES.

Las variables se sitúan en la ram, se deben declarar antes de ser utilizadas, se debe indicar el tipo de variable, luego el nombre y finalmente el valor inicial, este último es opcional.

Ejemplo.

unsigned int8 *variable=0x2a;*

2.10.5. OPERADORES.

- **Operadores de asignación.**

+=	Asignación de suma (x+=y es lo mismo que x=x+y)
-=	Asignación de resta (x-=y es lo mismo que x=x-y)
*=	Asignación de multiplicación (x*=y es lo mismo que x=x*y)
/=	Asignación de división (x/=y es lo mismo que x=x/y)
%=	Asignación del resto de la división (x%=y es lo mismo que x=x%y)
<<=	Asignación de desplazamiento a la izquierda (x<<=y es igual que x=x<<y)
>>=	Asignación de desplazamiento a derecha (x>>=y es igual que x=x>>y)
&=	Asignación AND de bits (x&=y es lo mismo que x=x&y)
|=	Asignación OR de bits (x|=y es lo mismo que x=x|y)
^=	Asignación OR EXCLUSIVA de bits (x^=y es lo mismo que x=x^y)

- **Operadores aritméticos.**

+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo, resto de una división entera
--	Incremento
++	Decremento
sizeof	Determina el tamaño, en bytes, de un operando

- **Operadores relacionales.**

<	Menor que
>	Mayor que
>=	Mayor o igual que
<=	Menor igual que
==	Igual
!=	Distinto
?:	Expresión condicional

- **Operadores lógicos.**

!	NOT
&&	AND
||	OR

- Operadores de bits.

~	Complemento a 1
&	AND
^	OR EXCLUSIVA
|	OR
>>	Desplazamiento a derechas
<<	Desplazamiento a izquierdas

2.10.6. FUNCIONES.

Una función puede devolver un valor a la sentencia que la ha llamado. El tipo de dato se indica en la definición de la función, en caso de no devolver un valor se debe especificar el valor VOID. La función, además de devolver un valor puede recibir parámetros. Para que la función devuelva un valor se usa la sentencia RETURN.

Ejemplo.

```
float cuadrado(float b)
{
  float c;
  c=b*b;
  return(c);
}
void main()
{
  float a;
  a=cuadrado(2.32);
}
```

2.10.7. DIRECTIVAS.

Las directivas las usuales son:

#fuses opciones, permite definir la palabra de configuración para la programación del PIC.

Las opciones pueden ser.

fuses: pll1,pll2,pll3,pll4,pll5,pll6,pll10,pll12,cpudiv1,cpudiv2
fuses: cpudiv3,cpudiv4,nousbdiv,usbdiv,xt,xtpll,ec_io,ec,ecpll_io
fuses: ecpll,intrc_io,intrc,intxt,inths,hs,hspll,nofcmen,fcmen
fuses: noieso,ieso,put,noput,nobrownout,brownout_sw,brownout_nosl
fuses: brownout,borv45,borv43,borv27,borv21,novregen,vregen,nowdt
fuses: wdt,wdt1,wdt2,wdt4,wdt8,wdt16,wdt32,wdt64,wdt128,wdt256
fuses: wdt512,wdt1024,wdt2048,wdt4096,wdt8192,wdt16384,wdt32768
fuses: ccp2b3,ccp2c1,nopbaden,pbaden,nolpt1osc,lpt1osc,nomclr,mclr
fuses: nostvren,stvren,nolvp,lvp,icsp1,icsp2,noxinst,xinst,debug
fuses: nodebug,protect,noprotect,cpb,nocpb,cpd,nocpd,wrt,nowrt,wrtc
fuses: nowrtc,wrtb,nowrtb,wrtd,nowrtd,ebtr,noebtr,ebtrb,noebtrb
#use delay(clock=velocidad), permite definir la frecuencia del oscilador del PIC, el compilador lo utiliza para realizar los cálculos de tiempo.

#asm
Instrucciones
#endasm, permite ingresar instrucciones en ensamblador dentro del programa en C.

2.11. SENSOR DE MOVIMIENTO INFRARROJO (PIR).

Los detectores PIR (Passive Infrared) o Pasivo Infrarrojo, reaccionan sólo ante determinadas fuentes de energía tales como el calor del cuerpo humano o animales. Básicamente reciben la variación de las radiaciones infrarrojas del medio ambiente que cubre. Es llamado pasivo debido a que no emite radiaciones, sino que las recibe. Estos captan la presencia detectando la diferencia entre el calor emitido por el cuerpo humano y el espacio alrededor.

Su componente principal es el sensor piroeléctrico. Se trata de un componente electrónico diseñado para detectar cambios en la radiación infrarroja recibida. Generalmente dentro de su encapsulado incorporan un transistor de efecto de campo que amplifica la señal eléctrica que genera cuando se produce dicha variación de radiación recibida.

La información infrarroja llega al sensor piroeléctrico a través de una lente de fressnell que divide el área protegida en sectores. Se distribuyen lentes con diferentes características: gran angular, cortina, corredor, antimascotas, etc.

Cuando se instala un sensor infrarrojo (PIR) y se lo energiza por primera vez, este comenzara a “acostumbrarse” a la radiación infrarroja del ambiente (todos los cuerpos que están a una temperatura superior a 0° Kelvin, emiten radiación infrarroja; esta radiación infrarroja aumenta si

aumenta la temperatura del cuerpo en cuestión. Es decir, que recibe la información infrarroja de una pared, el piso, los muebles, etc).

Una vez que se mantiene estable, si un intruso ingresa al recinto se experimentará un cambio en la radiación infrarroja del ambiente y el PIR dará una condición de alarma.

Con objeto de lograr total confiabilidad, esta tecnología integra además, un filtro especial de luz que elimina toda posibilidad de falsas detecciones causadas por la luz visible (rayos solares), así como circuitos especiales que dan mayor inmunidad a ondas de radio frecuencia.

2.12. SENSOR DE RUPTURA DE VIDRIOS.

Un sensor de ruptura de vidrios consta de un micrófono, conectado a un circuito electrónico sensible a las altas frecuencias provocadas por la fractura de un vidrio (el chasquido previo).

Cuando una ventana o puerta de vidrio se rompe, unos cuantos milisegundos antes, produce una onda expansiva de bajísima frecuencia, producto de la flexión del material. Si se golpea suavemente la ventana con el puño, se genera ese sonido. El mismo está siempre presente como parte de la secuencia de sonidos asociados con la rotura del cristal. Por tanto, los sensores de ruptura de vidrios procesaran el sonido en secuencia:

El sonido grave proveniente de la flexión del vidrio.

El sonido agudo producto de la fractura misma.

2.13. SENSOR DE IMPACTO.

El sensor de impacto utiliza un dispositivo piezoeléctrico que detecta las vibraciones y las convierte en señal alterna, esta señal para por una serie de amplificadores y filtros de modo que cuando sucede el impacto se genera un nivel bajo (0v.).

2.14. SENSOR MAGNÉTICO PARA PUERTAS.

Consta de dos partes, una es un imán permanente y la otra es un switch que se cierra con la presencia de un campo magnético.

2.15. CONVERTIDOR STEP DOWN (BUCK O REDUCTOR). MODO DE OPERACIÓN CONTINÚA.

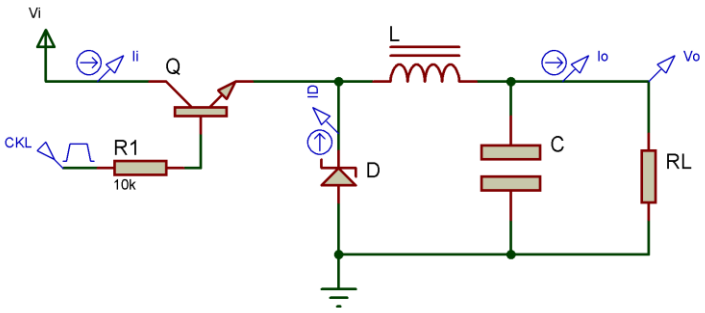


Figura 0.38 Circuito básico de un convertidor Step Down.

La señal rectangular CKL es una onda rectangular cuya relación $\frac{ton}{T} = \delta: ciclo\ de\ trabajo$, de modo que:

$$ton = \delta * T \quad , \quad toff = (1 - \delta) * T \quad y \quad T = ton + toff$$

2.15.1. CORRIENTE PROMEDIO I.

La corriente promedio I está definida como:

$$I = \frac{1}{T}\int_0^T i(t) * dt => I = \frac{1}{T} * [área\ bajo\ la\ curva]$$

2.15.2. CORRIENTE PROMEDIO EN EL TRANSISTOR Q.

Durante el tiempo *ton* del periodo *T*, el transistor se pone en estado de saturación cargando al inductor *L*, el diodo *D* se polariza inversamente y queda como circuito abierto.

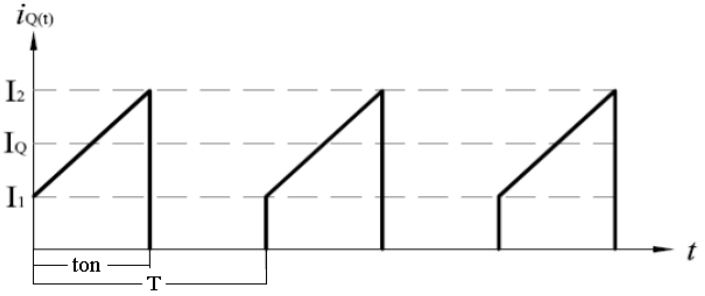


Figura 0.39 Forma de onda de corriente en el transistor Q.

$$IQ = \frac{1}{T}\int_0^{ton} iQ(t) * dt => IQ = \frac{1}{T} * [Área\ de\ trapezoide]$$

$$IQ = \frac{1}{T} * \left(\frac{I2 + I1}{2}\right) * ton \implies IQ = \delta * \frac{I2 + I1}{2}$$

2.15.3. CORRIENTE PROMEDIO EN EL DIODO D.

Durante el tiempo *toff* del periodo *T,* el transistor se pone en estado de corte, el diodo *D* se polariza directamente dejando circular la corriente almacenada en el inductor *L*.

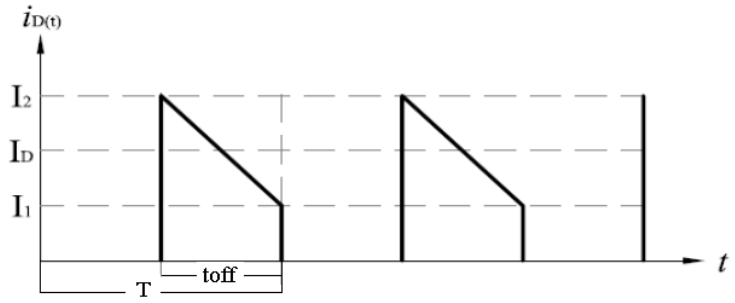


Figura 0.40 Forma de onda de corriente en el diodo D.

$$ID = \frac{1}{T}\int_0^{toff} iD(t) * dt \implies ID = \frac{1}{T} * [\text{Área de trapezoide}]$$

$$ID = \frac{1}{T} * \left(\frac{I2 + I1}{2}\right) * toff \implies ID = (1 - \delta) * \frac{I2 + I1}{2}$$

2.15.4. CORRIENTE PROMEDIO EN EL INDUCTOR L.

En el inductor *L* la corriente circula durante todo el periodo *T* (condición para trabajar en modo continuo).

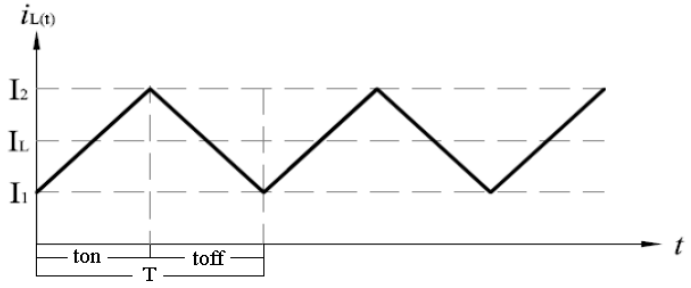


Figura 0.41 Forma de onda de corriente por el inductor L.

$$IL = IQ + ID \implies IL = \frac{I2 + I1}{2}$$

El incremento de la corriente en el inductor *L* durante el tiempo *ton* será:

$$\Delta iL(+) = \frac{1}{L} * \int_0^{ton} vL(t) * dt \;\; => \;\; \Delta iL(+) = \frac{1}{L} * \int_0^{ton} (Vi - VQsat - Vo) * dt$$

$$\Delta iL(+) = \frac{1}{L} * (Vi - VQsat - Vo) * ton$$

El decremento de la corriente en el inductor *L* durante el tiempo *tof* será:

$$\Delta iL(-) = \frac{1}{L} * \int_0^{toff} vL(t) * dt \;\; => \;\; \Delta iL(-) = \frac{1}{L} * \int_0^{toff} (VD + Vo) * dt$$

$$\Delta iL(-) = \frac{1}{L} * (VD + Vo) * toff$$

En régimen permanente $\Delta iL(+) = \Delta iL(-)$

$$\frac{1}{L} * (Vi - VQsat - Vo) * ton = \frac{1}{L} * (VD + Vo) * toff$$

$$(Vi - VQsat) * ton - VD * toff = Vo * (ton + toff)$$

$$(Vi - VQsat + VD) * ton - VD * T = Vo * T$$

$$Vo = \frac{ton}{T} * (Vi - VQsat + VD) - VD$$

$$Vo = \delta * (Vi - VQsat + VD) - VD$$

Para transistores y diodos ideales: *VQsat=0 y VD=0.*

$$Vo = \delta * Vi$$

La corriente que circula por el transistor *Q* es la misma que entrega la fuente de alimentación *Vi* y la corriente *IL* es la misma corriente que se entrega a la carga *Io.*

$$IQ = Ii = \delta * \frac{I2 + I1}{2} \quad y \quad IL = Io = \frac{I2 + I1}{2}$$

$$Io = \frac{Ii}{\delta}$$

2.15.5. POTENCIA DISIPADA POR EL TRANSISTOR.

$$PQ = VceSat * \delta * Io + \frac{1}{2} fs * Vi * \delta * Io * (ton + toff)$$

fs es la frecuencia del oscilador, *ton* y *toff* son los tiempos de encendido y apagado del transistor(Se obtener de la hoja de datos del transistor)

2.15.6. POTENCIA DISIPADA POR EL DIODO.

$$PD = VD * (1 - \delta) * Io + \frac{1}{2} fs * Vi * (1 - \delta) * Io * (ton + toff)$$

fs es la frecuencia del oscilador, *ton* y *toff* son los tiempos de encendido y apagado del diodo(Se obtener de la hoja de datos del transistor)

2.15.7. INDUCTANCIA MÍNIMA.

Dentro de la operación en modo continuo existe un valor inductancia mínimo que garantiza el trabajo en este tipo de operación y se le denomina *Lmin*, esto también sugiere que hay un valor mínimo de corriente que debe circular por la carga y se denomina *Iomin*.

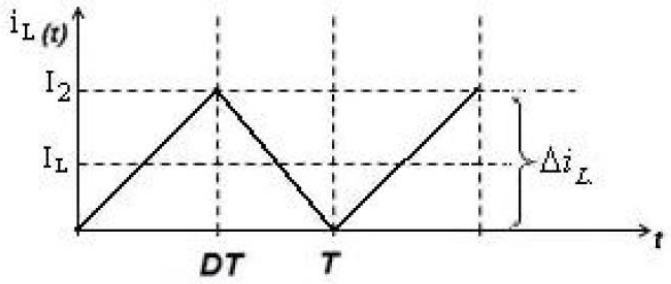


Figura 0.42 Forma de onda en el inductor L para la inductancia mínima.

El voltaje en la bobina está dado por:

$$VL = L * \frac{diL(t)}{dt} \Rightarrow \Delta VL = L * \frac{\Delta IL}{\Delta t}$$

Para el caso extremo tenemos:

$$IL = Iomin \ , \quad I1 = 0 \quad , \quad I2 = \Delta IL$$

$$I2 = 2 * IL => \ I2 = 2 * Iomin => \Delta IL = 2 * Iomin$$

De la ecuación del voltaje en el inductor.

$$\Delta VL = Lmin * \frac{2 * Iomin}{\delta * T}$$

$$Vi - Vo = Lmin * \frac{2 * Iomin}{\delta * T} => Lmin = \frac{(Vi - Vo) * \delta * T}{2 * Iomin}$$

$$Lmin = \frac{(Vi - Vo) * \delta}{2 * Iomin * Fs}, \quad Fs = \frac{1}{T} = frec\ del\ conmutador.$$

2.15.8. DIMENSIONAMIENTO DEL CAPACITOR DE SALIDA.

El nivel de rizado en la salida se define por dos parámetros, el rizado de provocado por el capacitor y el provocado por la resistencia equivalente en serie (ERS).

$$VoRPP = VRR + VRC$$

Donde VRR representa el rizado en la resistencia serie equivalente y VRC representa el rizado propio del capacitor.

En este punto se debe utilizar un criterio de diseño para la elección del valor de la capacitancia y de ESR. Se puede considerar que el rizado de la señal de salida está provocado fundamentalmente por el efecto del capacitor propiamente.

Si:

$$VRR = 0.1 * VRC => VoRPP \approx VRC$$

$$iC = C * \frac{dVC}{dt} => \Delta iC = C * \frac{\Delta VC}{\Delta t}$$

Durante el ciclo de carga del capacitor se tiene.

$$\Delta VC = VoRPP, \quad \Delta i = IOmax, \quad \Delta t = \delta T = \frac{\delta}{Fs}\ ;\ C = \frac{IOmax * \delta}{Fs * VoRPP}$$

2.15.9. RESUMEN DE ECUACIONES DE DISEÑO.

$$Vo = \delta * Vi \qquad ton = \delta T \;\; => ton = \frac{\delta}{Fs}$$

$$Io = \frac{Ii}{\delta}$$

$$IQ = \delta * Io$$

$$ID = (1 - \delta) * Io$$

$$PQ = VceSat * \delta * Io + \frac{1}{2} fs * Vi * \delta * Io * (ton + toff)$$

$$PD = VD * (1 - \delta) * Io + \frac{1}{2} fs * Vi * (1 - \delta) * Io * (ton + toff)$$

$$Lmin = \frac{(Vi - Vo) * \delta}{2 * Iomin * Fs}, \qquad I2 - I1 = \frac{(Vi - Vo) * \delta}{L * Fs}, \qquad IL = Io = \frac{I2 + I1}{2}$$

$$C = \frac{Iomax * \delta}{Fs * VoRPP}$$

2.15.10. REGULADOR DE VOLTAJE CONMUTADO MC34063A STEP DOWN.

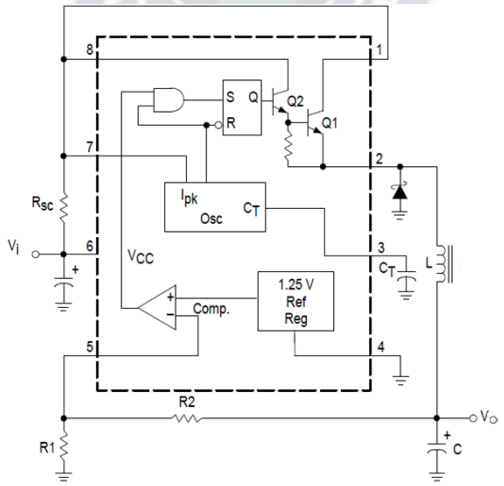


Figura 0.43 Circuito de aplicación del chip MC34063A.

2.15.11. ECUACIONES RELEVANTES PARA EL REGULADOR MC34063.

$$CT = 4x10^{-5} * ton, \qquad RSC = \frac{0.3}{Imax}, \qquad Vo = 1.25 * \left(1 + \frac{R2}{R1}\right)$$

CAPITULO III

INGENIERÍA DEL PROYECTO.

Dentro del domicilio se encuentra el hardware de monitoreo verificando los estados de las puertas, ventanas y sensores instalados emitiendo periódicamente un reporte hacia un servidor para que el personal de monitoreo pueda efectuar alguna acción según los eventos que reciban, adicionalmente el cliente dueño del domicilio tendrá una página web donde podrá ver el estado se su domicilio.

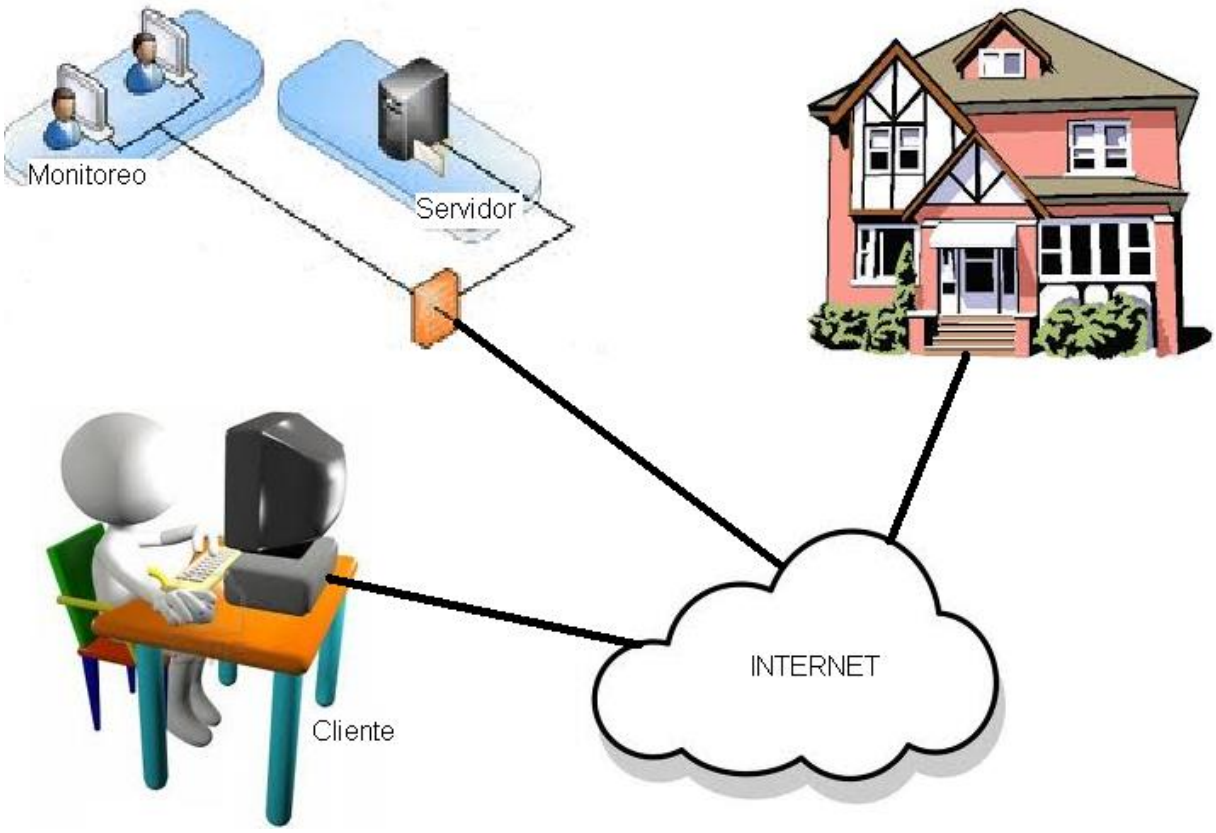


Figura 0.1 Modelo básico del proyecto.

3.1. BASE DE DATOS.

La base de datos será la que contenga toda la información sobre el cliente como ser: nombres de usuarios, contraseñas y el estado del domicilio u oficina, etc.

La base de datos se implementara sobre Access de Microsoft 2010 cuyo nombre será ***clientes1.accdb,*** constara de cinco tablas llamadas ***alias_estados, alias_zonas, estados, historial y usuarios.***

La tabla ***alias_estados*** contendrá los siguientes campos.

alias_estados		
zonas	**abierto**	**cerrado**
zona1	Abierto	cerrado
zona2	Abierto	cerrado
zona3	Abierto	cerrado
zona4	Abierto	cerrado
zona5	Abierto	cerrado
zona6	Abierto	cerrado
zona7	Abierto	cerrado
zona8	Abierto	cerrado
salida1	Abierto	cerrado
salida2	Abierto	cerrado
salida3	Abierto	cerrado
salida4	Abierto	cerrado

La tabla ***alias_zonas*** tendrá los siguientes campos.

alias_zonas											
zona0	**zona1**	**zona2**	**zona3**	**zona4**	**zona5**	**zona6**	**zona7**	**salida0**	**salida1**	**salida2**	**salida3**
zona0	zona1	zona2	zona3	zona4	zona5	zona6	zona7	pgm0	pgm1	pgm2	pgm3

La tabla ***estados*** tendrá los siguientes campos.

estados									
codigo_cliente	**cliente**	**zona0**	**zona1**	**zona2**	**zona3**	**zona4**	**zona5**	**zona6**	**zona7**
123	zabaleta	abierto	abierto	abierto	abierto	abierto	abierto	abierto	abierto
456	flores	abierto	abierto	abierto	abierto	abierto	abierto	abierto	abierto
789	murillo	abierto	abierto	abierto	abierto	abierto	abierto	abierto	abierto

estados					
evento_zona0	**evento_zona1**	**evento_zona2**	**evento_zona3**	**evento_zona4**	**evento_zona5**
no_disparado	no_disparado	no_disparado	no_disparado	no_disparado	no_disparado

estados					
evento_zona0	evento_zona1	evento_zona2	evento_zona3	evento_zona4	evento_zona5
no_disparado	no_disparado	no_disparado	no_disparado	no_disparado	no_disparado
no_disparado	no_disparado	no_disparado	no_disparado	no_disparado	no_disparado

estados						
evento_zona6	evento_zona7	salida0	salida1	salida2	salida3	alarma
no_disparado	no_disparado	activado	activado	activado	activado	desactivada
no_disparado	no_disparado	desactivado	desactivado	desactivado	desactivado	modo_presencia
no_disparado	no_disparado	desactivado	desactivado	desactivado	desactivado	desactivada

estados						
id_equipo	numero_equipo	numero_socket	comando	usuario_activo1	usuario_activo2	usuario_activo3
smdu0001	70660715	-1	-	-	-	-
smdu0002	70660715	-1	-	-	-	-
smdu0003	70660715	-1	-	-	-	-

La tabla ***historial*** tendrá los siguientes campos.

historial					
Id	cliente	codigo_cliente	nick	accion	fecha_accion
1	zabaleta	123	jose	activar salida0	26-05-2016 14:31:11
2	zabaleta	123	jose	activar salida0	26-05-2016 14:33:31

La tabla ***usuarios*** tendrá los siguientes campos.

usuarios						
codigo_usuario	nick	usuario	ci	telefono	enviar_evento	codigo_cliente
963	jose	jose				123
852	rique	enrrique				123
741	jere	jeremias				123
951	edu	edwin				456
357	lucho	luis				456

3.2. PROGRAMA DE MONITOREO.

El programa de monitoreo será operado por los vigilantes, estos vigilantes tendrán a su cargo el monitoreo, en caso de la detección de intrusos coordinaran el auxilio a los domicilios u oficinas clientes.

El programa es implementado en Microsoft Visual Basic 6.0.

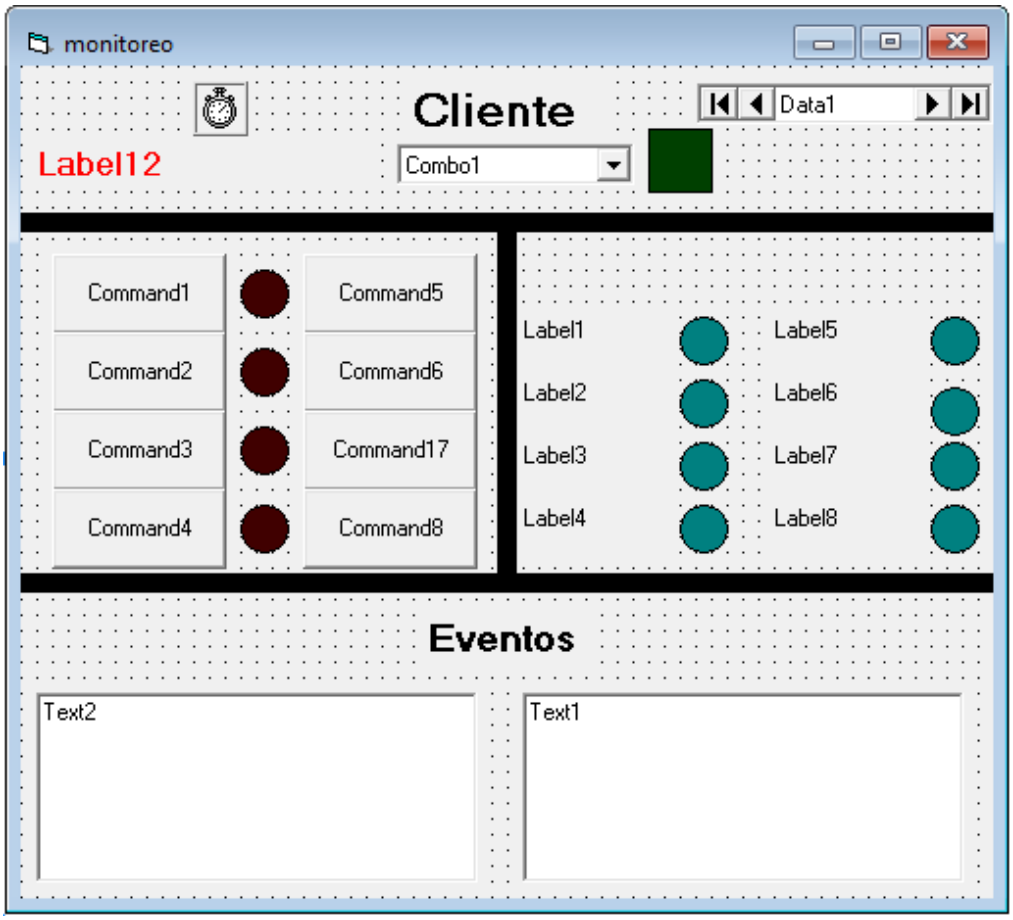


Figura 0.2 Formulario de para el programa de monitoreo.

Variables globales.

```
Dim Wrk As Workspace        'se crea un espacio de trabajo
Dim MIWRK As Database       'se crea un base de datos
Dim MIrec As Recordset      'se crea una tabla
Dim lstConect As String     'definición de una variable localstringconect
Dim MIsql As String         'variable para contener los comandos SQL
Dim error As Integer        'variable bandera que señaliza un error
```

Evento load.
Lee la lista de clientes de la base de datos y los carga en el combo1.

```
Private Sub Form_Load()
abrir_db                                'Abre la base de datos
consulta ("SELECT cliente FROM estados")
Do Until MIrec.EOF                      'Mientras no llegue al final del registro
   Combo1.AddItem (MIrec("cliente"))    'Adiciona al Combobox1 los resultados de la consulta
   MIrec.MoveNext                       'Apunta al siguiente registro
Loop
cerrar_db                               'Cierra la base de datos
End Sub
```

Evento Command1.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para activar la *salida0* del equipo de monitoreo.

```
Private Sub Command1_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "01" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command2.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para activar la *salida1* del equipo de monitoreo.

```
Private Sub Command2_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "11" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command3.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para activar la *salida2* del equipo de monitoreo.

```
Private Sub Command3_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
```

```
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "21" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command4.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para activar la *salida3* del equipo de monitoreo.

```
Private Sub Command4_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "31" & "' ‘‘&
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command5.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para desactivar la *salida0* del equipo de monitoreo.

```
Private Sub Command5_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "00" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command6.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para desactivar la *salida1* del equipo de monitoreo.

```
Private Sub Command6_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "10" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command7.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para desactivar la *salida2* del equipo de monitoreo.

```
Private Sub Command7_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "20" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento Command8.
Obtiene el ID del equipo asignado al cliente y actualiza la base de datos con el comando para desactivar la *salida3* del equipo de monitoreo.

```
Private Sub Command8_Click()
abrir_db
consulta ("SELECT id_equipo FROM estados WHERE cliente=" & "'" & Combo1.Text & "'")
consulta ("UPDATE estados SET comando =" & "'" & MIrec("id_equipo") & "30" & "' " &
"WHERE cliente=" & "'" & Combo1.Text & "'")
cerrar_db
End Sub
```

Evento timer1.
Este evento sucede cada segundo. Verifica si se seleccionó a algún usuario del combo box1.

Efectúa una consulta a la tabla *alias_zonas* de la base de datos para asignar las etiquetas de nombres a los botones *command* y a los *labels.*

Consulta la tabla *estados* para verificar si el equipo asignado al cliente seleccionado está conectado al servidor y actualiza los estados en el formulario de monitoreo.

Lee y muestra los eventos sucedidos en los equipos de los usuarios.

```
Private Sub Timer1_Timer()
Dim i As Integer
If (Combo1.Text <> "") Then
   abrir_db
   If error = 1 Then
      error = 0
      Exit Sub
   End If
```

```
   consulta ("SELECT * FROM alias_zonas")
   Label1.Caption = MIrec("zona0")
   Label2.Caption = MIrec("zona1")
   Label3.Caption = MIrec("zona2")
   Label4.Caption = MIrec("zona3")
   Label5.Caption = MIrec("zona4")
   Label6.Caption = MIrec("zona5")
   Label7.Caption = MIrec("zona6")
   Label8.Caption = MIrec("zona7")
   Command1.Caption = "Activar " & MIrec("salida0")
   Command2.Caption = "Activar " & MIrec("salida1")
   Command3.Caption = "Activar " & MIrec("salida2")
   Command4.Caption = "Activar " & MIrec("salida3")
   Command5.Caption = "Desactivar " & MIrec("salida0")
   Command6.Caption = "Desactivar " & MIrec("salida1")
   Command7.Caption = "Desactivar " & MIrec("salida2")
   Command8.Caption = "Desactivar " & MIrec("salida3")
'*********************************************************************
   consulta ("SELECT * FROM estados WHERE cliente= " & "'" & Combo1.Text & "'")
   MIrec.MoveFirst
'*********************************************************************
   If (MIrec("numero_socket") <> "-1") Then
      Shape13.BackColor = &HFF00&
   Else
      Shape13.BackColor = &H4000&
   End If
'*********************************************************************
   If (MIrec("alarma") = "desactivada") Then
      Label12.Caption = "Alarma desactivada"
   ElseIf (MIrec("alarma") = "activada") Then
      Label12.Caption = "Alarma activada"
   Else
      Label12.Caption = "Modo presencia"
   End If
'*********************************************************************
   If (MIrec("zona0") = "abierto") Then
      Shape5.BackColor = &HFF00FF
   Else
      Shape5.BackColor = &H808000
   End If
'*********************************************************************
   If (MIrec("zona1") = "abierto") Then
      Shape6.BackColor = &HFF00FF
   Else
      Shape6.BackColor = &H808000
   End If
'*********************************************************************
```

```
   If (MIrec("zona2") = "abierto") Then
      Shape7.BackColor = &HFF00FF
   Else
      Shape7.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("zona3") = "abierto") Then
      Shape8.BackColor = &HFF00FF
   Else
      Shape8.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("zona4") = "abierto") Then
      Shape9.BackColor = &HFF00FF
   Else
      Shape9.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("zona5") = "abierto") Then
      Shape10.BackColor = &HFF00FF
   Else
      Shape10.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("zona6") = "abierto") Then
      Shape11.BackColor = &HFF00FF
   Else
      Shape11.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("zona7") = "abierto") Then
      Shape12.BackColor = &HFF00FF
   Else
      Shape12.BackColor = &H808000
   End If
'*************************************************************************
   If (MIrec("salida0") = "activado") Then
      Shape1.BackColor = &HFF&
   Else
      Shape1.BackColor = &H40&
   End If
'*************************************************************************
   If (MIrec("salida1") = "activado") Then
      Shape2.BackColor = &HFF&
   Else
      Shape2.BackColor = &H40&
   End If
```

```
'*********************************************************************
   If (MIrec("salida2") = "activado") Then
      Shape3.BackColor = &HFF&
   Else
      Shape3.BackColor = &H40&
   End If
'*********************************************************************
   If (MIrec("salida3") = "activado") Then
      Shape4.BackColor = &HFF&
   Else
      Shape4.BackColor = &H40&
   End If
'*********************************************************************
   Text1.Text = ""
   If (MIrec("evento_zona0") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label1.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona1") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label2.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona2") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label3.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona3") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label4.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona4") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label5.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona5") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label6.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona6") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label7.Caption & vbCrLf
   End If
'*********************************************************************
   If (MIrec("evento_zona7") = "disparado") Then
      Text1.Text = Text1.Text & "Evento en " & Label8.Caption & vbCrLf
   End If
```

```
'*************************************************************************
'Muestra si hubo algún evento en los clientes
'*************************************************************************
   Text2.Text = ""
   consulta ("SELECT
cliente,evento_zona0,evento_zona1,evento_zona2,evento_zona3,evento_zona4,evento_zona5,eve
nto_zona6,evento_zona7 FROM estados")
   MIrec.MoveFirst
   Do Until MIrec.EOF
      For i = 0 To 7 Step 1
         If (MIrec("evento_zona" & i)) = "disparado" Then
            Text2.Text = Text2.Text & (MIrec("cliente")) & vbCrLf
            Exit For
         End If
      Next
      MIrec.MoveNext
   Loop
   cerrar_db
End If
End Sub
'*************************************************************************
```

Subrutinas para el manejo de la base de datos.

```
'*************************************************************************
Sub abrir_db()
On Error GoTo error_db:
Set Wrk = CreateWorkspace("clientes1", "", "", dbUseODBC)          'SETEO ESPACIO DE
TRABAJO (nombre DE BASE DE DATOS,usuario,contraseña,TIPO DE ENLACE)
lstConect = "ODBC;UID=;PWD=;eventos"
Set MIWRK = Wrk.OpenDatabase("clientes1", dbDriverNoPrompt, False, lstConect)
error_db: If Err.Number <> 0 Then
         error = 1
       End If
End Sub
'*************************************************************************
Sub cerrar_db()
MIrec.Close
MIWRK.Close
Wrk.Close
End Sub
'*************************************************************************
Sub consulta(MIsql)
Set MIrec = MIWRK.OpenRecordset(MIsql, dbOpenSnapshot)
End Sub
'*************************************************************************
```

3.3. PÁGINA WEB PARA EL MONITOREO POR PARTE DEL CLIENTE.

La Página web está diseñada para que el cliente pueda tener constancia de los eventos sucedidos en su domicilio u oficina. La Página web está construida sobre HTML y PHP.

3.3.1. INICIO DE LA PAGINA WEB (INICIO.HTML).

Figura 0.3 Página de inicio.

Se encarga de mostrar el interface de inicio y recibir los datos del usuario.

```
<html>
<body bgColor=lightcyan>
<embed src=rata.wav loop="true" hidden="true"  style="TOP: 162px; LEFT: 10px"
autostart="true">
<hr>
<center>
        <b style="FONT-SIZE: large">
                <font color=forestgreen size=7>
                        Vigilamos para tu seguridad
                </font>
        </B>
</center>
<IMG src="monster3.gif">
<center>
        <b style="BACKGROUND-REPEAT: repeat; FONT-SIZE: large">
                <font color=darkgreen size=6>
                        Para Acceder ingrese codigo
                </font>
        </b>
</center>
<hr>
<form name="usuario" method="post" action="inicio.php">
        Nombre:<INPUT name="nick">
        Password:<INPUT type=password name="codigo_usuario">
```

```
        <INPUT type="submit" value="ingresar">
</form>
<hr>
</body>
</html>
```

3.3.2. PÁGINA DE AUTENTICACIÓN DEL PARA EL USUARIO (INICIO.PHP).

Se encarga de autenticar los datos introducidos por el usuario.

```
<?php
$nick=$_POST['nick'];
$codigo_usuario=$_POST['codigo_usuario'];
$db='clientes1';
$usu='';
$pas='';
$db=odbc_connect($db,$usu,$pas);
if ($db)
{
   $resultado=odbc_exec($db,sprintf("SELECT nick FROM usuarios WHERE
codigo_usuario='%s'",$codigo_usuario));
   if ($resultado)
   {
      $valor = odbc_fetch_array($resultado);
      if($valor['nick']==$nick)
      {
         odbc_free_result($resultado);
         echo ("usuario registrado");
         $resultado=odbc_exec($db,sprintf("SELECT codigo_cliente FROM usuarios WHERE
codigo_usuario='%s'",$codigo_usuario));
         $valor=odbc_fetch_array($resultado);
         $codigo_cliente=$valor['codigo_cliente'];
         odbc_free_result($resultado);
         $resultado=odbc_exec($db,sprintf("SELECT cliente FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente));
         $valor=odbc_fetch_array($resultado);
         $cliente=$valor['cliente'];
         odbc_free_result($resultado);
         $resultado=odbc_exec($db,sprintf("SELECT
usuario_activo1,usuario_activo2,usuario_activo3 FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente));
         $valor = odbc_fetch_array($resultado);
if(($valor['usuario_activo1']==$nick)||($valor['usuario_activo2']==$nick)||($valor['usuario_ac
tivo3']==$nick))
         {
            odbc_free_result($resultado);
            odbc_close($db);
```

```
            printf("El usuario %s ya tiene una sesión abierta",$nick);
?>
            <meta http-equiv="Refresh" content="2;URL=http://localhost/inicio/inicio.html">
<?php
        }
        else
        {
            odbc_free_result($resultado);
            if($valor['usuario_activo1']=='-')
            {
                $consulta5=sprintf("UPDATE estados SET usuario_activo1='%s' WHERE
codigo_cliente='%s'",$nick,$codigo_cliente);
                odbc_exec($db,$consulta5);
            }
            elseif($valor['usuario_activo2']=='-')
            {
                $consulta6=sprintf("UPDATE estados SET usuario_activo2='%s' WHERE
codigo_cliente='%s'",$nick,$codigo_cliente);
                odbc_exec($db,$consulta6);
            }
            elseif($valor['usuario_activo3']=='-')
            {
                $consulta7=sprintf("UPDATE estados SET usuario_activo3='%s' WHERE
codigo_cliente='%s'",$nick,$codigo_cliente);
                odbc_exec($db,$consulta7);
            }
            require_once 'Request.php';
            $request = new Request();
            $request->addParam("cliente",$cliente);
            $request->addParam("codigo_cliente",$codigo_cliente);
            $request->addParam("nick",$nick);
            $request->addParam("salir",0);
            $request->forward("monitorear.php");
        }
    }
    else
    {
        odbc_free_result($resultado);
        odbc_close($db);
        print("No esta autorizado para ver esta pajina");
?><meta http-equiv="Refresh" content="1;URL=http://localhost/inicio/inicio.html"><?php
    }
  }
  Else{odbc_free_result($resultado);odbc_close($db);print("Error en la consulta");}
}
else{print("<p>Imposible conectarse con la base de datos/p>");}
?>
```

3.3.3. PÁGINA DE MONITOREO (MONITOREO.PHP).

Se encarga de mostrar al usuario los estados de las entradas y salidas del equipo de monitoreo.

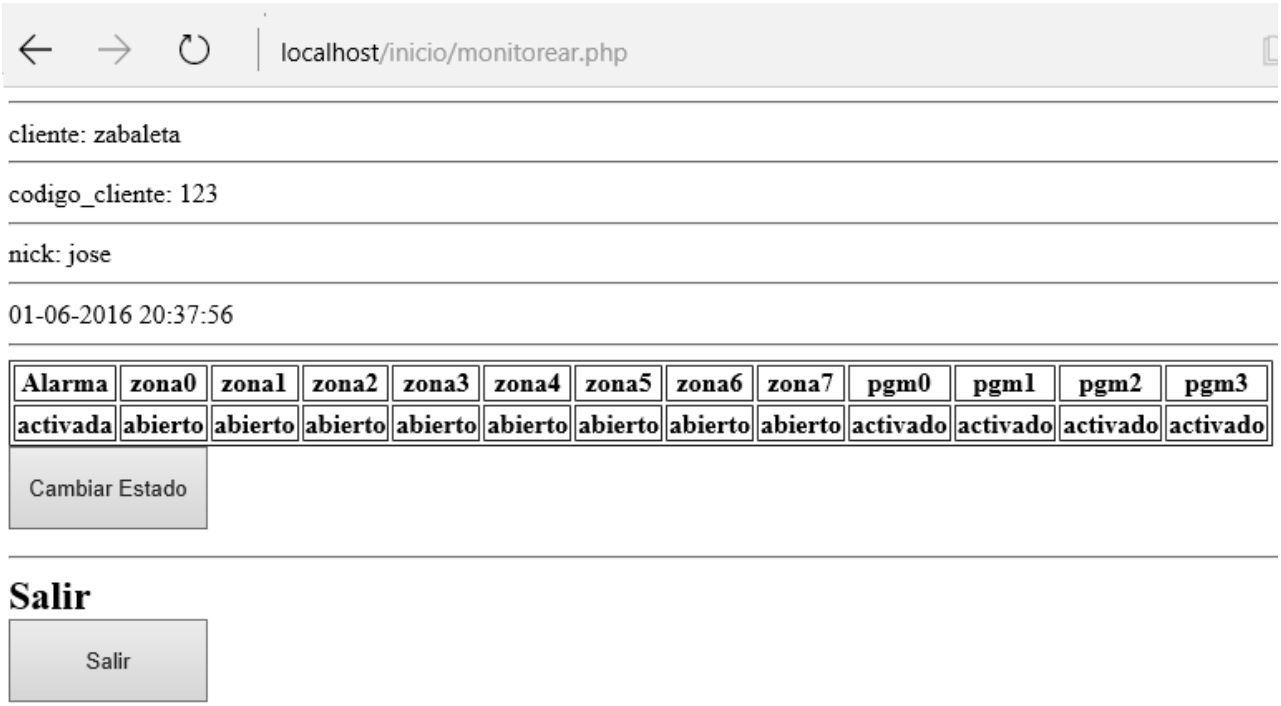


Figura 0.4 Página de monitoreo.

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$salir=$_POST['salir'];
if(!$cliente||!$codigo_cliente)
{
   print("No está autorizado para ver esta página");
?>
   <meta http-equiv="Refresh" content="1;URL=http://localhost/inicio/inicio.html">
<?php
   exit();
}
$base_de_datos='clientes1';$usu='';$pas='';
$db = odbc_connect($base_de_datos,$usu,$pas);
if (!$db)
{
   print "<p>Imposible conectarse con la base de datos/p>";
   exit();
}
```

```
else
{
   $consulta1=sprintf("SELECT usuario_activo1,usuario_activo2,usuario_activo3 FROM
estados WHERE codigo_cliente='%s'",$codigo_cliente);
   $resultado=odbc_exec($db,$consulta1);
   $valor = odbc_fetch_array($resultado);
   if($salir)
   {
      if($valor['usuario_activo1']==$nick)
      {
         $consulta1=sprintf("UPDATE estados SET usuario_activo1 ='-' WHERE
codigo_cliente='%s'",$codigo_cliente);
         $resultado=odbc_exec($db,$consulta1);
      }
      if($valor['usuario_activo2']==$nick)
      {
         $consulta1=sprintf("UPDATE estados SET usuario_activo2 ='-' WHERE
codigo_cliente='%s'",$codigo_cliente);
         $resultado=odbc_exec($db,$consulta1);
      }
      if($valor['usuario_activo3']==$nick)
      {
         $consulta1=sprintf("UPDATE estados SET usuario_activo3 ='-' WHERE
codigo_cliente='%s'",$codigo_cliente);
         $resultado=odbc_exec($db,$consulta1);
      }
      odbc_free_result($resultado);
      odbc_close($db);
      print("Secion finalizada.....");
?>
      <meta http-equiv="Refresh" content="1;URL=http://localhost/inicio/inicio.html">
<?php
      exit();
   }
if(($valor['usuario_activo1']!=$nick)&&($valor['usuario_activo2']!=$nick)&&($valor['usuario
_activo3']!=$nick))
    {
      exit();
    }
   echo("<hr>");print("cliente: $cliente");
   echo("<hr>");print("codigo_cliente: $codigo_cliente");
   echo("<hr>");print("nick: $nick");
   echo("<hr>");date_default_timezone_set('America/La_Paz');
   echo date("d-m-Y H:i:s");
   $consulta1=sprintf("SELECT * FROM alias_zonas");
   echo("<hr>");
```

```
   $resultado=odbc_exec($db,$consulta1);
   $valor=odbc_fetch_array($resultado);
   print("<table border=1
        <tr><th>Alarma</th><th>$valor[zona0]</th><th>$valor[zona1]</th>
            <th>$valor[zona2]</th><th>$valor[zona3]</th><th>$valor[zona4]</th>
            <th>$valor[zona5]</th><th>$valor[zona6]</th><th>$valor[zona7]</th>
            <th>$valor[salida0]</th><th>$valor[salida1]</th><th>$valor[salida2]</th>
            <th>$valor[salida3]</th>
        </tr>");
   odbc_free_result($resultado);
   $consulta1=sprintf("SELECT * FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
   $resultado=odbc_exec($db,$consulta1);
   $valor=odbc_fetch_array($resultado);
   print(" </tr>
             <th>$valor[alarma]</th> <th>$valor[zona0]</th><th>$valor[zona1]</th>
             <th>$valor[zona2]</th><th>$valor[zona3]</th><th>$valor[zona4]</th>
             <th>$valor[zona5]</th><th>$valor[zona6]</th><th>$valor[zona7]</th>
             <th>$valor[salida0]</th><th>$valor[salida1]</th><th>$valor[salida2]</th>
             <th>$valor[salida3]</th>
           </tr>");
   print("</table>");
   odbc_free_result($resultado);
   odbc_close($db);
?>
   <HTML>
     <BODY>
       <form action="cambiar_estado.php" method="post">
         <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
         <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
         <input type="hidden" name="nick" value="<?php echo $nick;?>">
         <input type="hidden" name="salir" value=0>
         <input type="submit" value="Cambiar Estado" style="WIDTH: 120px; HEIGHT:
50px" size=50>
       </form>
       <HR>
       <FONT size=5>
         <STRONG>Salir</STRONG>
       </FONT>
       <FORM action="monitorear.php" method=post id=form3 name=form3>
         <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
         <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
         <input type="hidden" name="nick" value="<?php echo $nick;?>">
         <INPUT type="hidden" name =salir value=1>
```

```
            <INPUT type="submit" value="Salir" name=button1 style="WIDTH: 120px;
HEIGHT: 50px" size=50>
          </FORM>
          <HR>
        </BODY>
    </HTML>
<?php
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
    sleep(1);
}
?>
```

3.3.4. PÁGINA DE CAMBIO DE ESTADO DE LAS SALIDAS DEL EQUIPO DE MONITOREO (CAMBIAR_ESTADO.PHP).

Figura 0.5 Página de cambio de estado.

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$salir=$_POST['salir'];
if(!$cliente||!$codigo_cliente)
{
   print("No esta autorizado para ver esta página");
   ?>
      <meta http-equiv="Refresh" content="1;URL=http://localhost/inicio/inicio.html">
   <?php
   exit();
}
$base_de_datos='clientes1';
$usu='';
$pas='';
$db = odbc_connect($base_de_datos,$usu,$pas);
if (!$db)
{
      print "<p>Imposible conectarse con la base de datos    /p>";
      exit();
}
$consulta1=sprintf("SELECT salida0,salida1,salida2,salida3 FROM alias_zonas");
$resultado=odbc_exec($db,$consulta1);
$valor = odbc_fetch_array($resultado);

$salida0=$valor['salida0'];$salida1=$valor['salida1'];$salida2=$valor['salida2'];
$salida3=$valor['salida3'];
odbc_free_result($resultado);
odbc_close($db);
?>
<HTML>
   <BODY>
      <HR>
      <FONT size=5>
         <STRONG>Regresar</STRONG>
      </FONT>
      <FORM action="monitorear.php" method=post id=form3 name=form3>
         <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
         <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
         <input type="hidden" name="nick" value="<?php echo $nick;?>">
         <INPUT type="hidden" name =salir value=0>
         <INPUT type="submit" value="Atras" name=button1 style="WIDTH: 120px; HEIGHT:
50px" size=50>
      </FORM>
      <HR>
```

```
<FONT size=5>
  <STRONG><?php echo($salida0);?></STRONG>
</FONT>
<FORM action="activar_salida0.php" method=post>
  <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
  <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
  <input type="hidden" name="nick" value="<?php echo $nick;?>">
  <input type="hidden" name="activar" value="activar">
  <INPUT type="submit" value="<?php echo("Activar $salida0");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<FORM action="desactivar_salida0.php" method=post>
  <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
  <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
  <input type="hidden" name="nick" value="<?php echo $nick;?>">
  <input type="hidden" name="desactivar" value="desactivar">
  <INPUT type="submit" value="<?php echo("Desactivar $salida0");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<HR>
<FONT size=5>
  <STRONG><?php echo($salida1);?></STRONG>
</FONT>
<FORM action="activar_salida1.php" method=post>
  <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
  <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
  <input type="hidden" name="nick" value="<?php echo $nick;?>">
  <input type="hidden" name="activar" value="activar">
  <INPUT type="submit" value="<?php echo("Activar $salida1");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<FORM action="desactivar_salida1.php" method=post>
  <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
  <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
  <input type="hidden" name="nick" value="<?php echo $nick;?>">
  <input type="hidden" name="desactivar" value="desactivar">
  <INPUT type="submit" value="<?php echo("Desactivar $salida1");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<HR>
<FONT size=5>
  <STRONG><?php echo($salida2);?></STRONG>
</FONT>
```

```
<FORM action="activar_salida2.php" method=post>
<input type="hidden" name="cliente" value="<?php echo $cliente;?>">
<input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
<input type="hidden" name="nick" value="<?php echo $nick;?>">
<input type="hidden" name="activar" value="activar">
<INPUT type="submit" value="<?php echo("Activar $salida2");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<FORM action="desactivar_salida2.php" method=post>
<input type="hidden" name="cliente" value="<?php echo $cliente;?>">
<input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
<input type="hidden" name="nick" value="<?php echo $nick;?>">
<input type="hidden" name="desactivar" value="desactivar">
<INPUT type="submit" value="<?php echo("Desactivar $salida2");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<HR>
<FONT size=5>
<STRONG><?php echo($salida3);?></STRONG>
</FONT>
<FORM action="activar_salida3.php" method=post>
<input type="hidden" name="cliente" value="<?php echo $cliente;?>">
<input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
<input type="hidden" name="nick" value="<?php echo $nick;?>">
<input type="hidden" name="activar" value="activar">
<INPUT type="submit" value="<?php echo("Activar $salida3");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<FORM action="desactivar_salida3.php" method=post>
<input type="hidden" name="cliente" value="<?php echo $cliente;?>">
<input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
<input type="hidden" name="nick" value="<?php echo $nick;?>">
<input type="hidden" name="desactivar" value="desactivar">
<INPUT type="submit" value="<?php echo("Desactivar $salida3");?>" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
</FORM>
<HR>
<FONT size=5>
<STRONG>Alarma</STRONG>
</FONT>
<FORM action="activar_alarma.php" method=post>
<input type="hidden" name="cliente" value="<?php echo $cliente;?>">
```

```
        <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
        <input type="hidden" name="nick" value="<?php echo $nick;?>">
        <input type="hidden" name="activar" value="activar">
        <INPUT type="submit" value="Activar Alarma" name=button1 style="WIDTH: 120px;
HEIGHT: 50px" size=50>
    </FORM>
    <FORM action="desactivar_alarma.php" method=post>
        <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
        <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
        <input type="hidden" name="nick" value="<?php echo $nick;?>">
        <input type="hidden" name="desactivar" value="desactivar">
        <INPUT type="submit" value="Desactivar Alarma" name=button1 style="WIDTH:
120px; HEIGHT: 50px" size=50>
    </FORM>
    <FORM action="activar_mp.php" method=post>
        <input type="hidden" name="cliente" value="<?php echo $cliente;?>">
        <input type="hidden" name="codigo_cliente" value="<?php echo
$codigo_cliente;?>">
        <input type="hidden" name="nick" value="<?php echo $nick;?>">
        <input type="hidden" name="activar_mp" value="activar_mp">
        <INPUT type="submit" value="Activar Alarma Modo Presente" name=button1
style="WIDTH: 120px; HEIGHT: 50px" size=50>
    </FORM>
    <HR>
  </BODY>
</HTML>
```

3.3.5. ACCIÓN ACTIVAR SALIDA0 (ACTIVAR_SALIDA0.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar=$_POST['activar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar=="activar")
{
        $db = odbc_connect($base_de_datos,$usu,$pas);
        if (!$db)
        {
                print "<p>Imposible conectarse con la base de datos      /p>";
                exit();
        }
```

```
        else
        {
    $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
    $resultado=odbc_exec($db,$consulta1);
    $valor = odbc_fetch_array($resultado);
    $id_equipo=$valor['id_equipo'];
    $consulta1=sprintf("UPDATE estados SET comando='%s01' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
    $resultado=odbc_exec($db,$consulta1);
            if (!$resultado)
            {
                exit("Error en la consulta");
            }
            else
            {
                echo "Comando Enviado";
            }
            date_default_timezone_set('America/La_Paz');
            $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar
salida0','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
            odbc_exec($db,$consulta1);
            odbc_free_result($resultado);
            odbc_close($db);
        }
        require_once 'Request.php';
        $request = new Request();
        $request->addParam("cliente",$cliente);
        $request->addParam("codigo_cliente",$codigo_cliente);
        $request->addParam("nick",$nick);
        $request->addParam("salir",0);
        $request->forward("monitorear.php");
}
else
{
        echo "Error";
}
?>
```

El hardware estará formado por dos partes, la primera será lo que es el circuito de control y la segunda será formada por un teclado y LCD que brindaran información del circuito de control al cliente.

3.3.6. ACCIÓN DESACTIVAR SALIDA0 (DESACTIVAR_SALIDA0.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$desactivar=$_POST['desactivar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($desactivar=="desactivar")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s00' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
      else
      {
         echo "Comando Enviado";
      }
      date_default_timezone_set('America/La_Paz');
      $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','desactivar
salida0','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
      odbc_exec($db,$consulta1);
      odbc_free_result($resultado);
      odbc_close($db);
   }
   require_once 'Request.php';
   $request = new Request();
```

```
   $request->addParam("cliente",$cliente);
   $request->addParam("codigo_cliente",$codigo_cliente);
   $request->addParam("nick",$nick);
   $request->addParam("salir",0);
   $request->forward("monitorear.php");
}
else
{
   echo "Error";
}
?>
```

3.3.7. ACCIÓN ACTIVAR SALIDA1 (ACTIVAR_SALIDA1.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar=$_POST['activar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar=="activar")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
         print "<p>Imposible conectarse con la base de datos    /p>";
         exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];
      $consulta1=sprintf("UPDATE estados SET comando='%s11' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
      else
      {
         echo "Comando Enviado";
```

```
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar
salida1','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else
{
    echo "Error";
}
?>
```

3.3.8. ACCIÓN DESACTIVAR SALIDA1 (DESACTIVAR_SALIDA1.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$desactivar=$_POST['desactivar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($desactivar=="desactivar")
{
    $db = odbc_connect($base_de_datos,$usu,$pas);
    if (!$db)
    {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
    }
    else
    {
        $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
        $valor = odbc_fetch_array($resultado);
```

```
        $id_equipo=$valor['id_equipo'];

        $consulta1=sprintf("UPDATE estados SET comando='%s10' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
        if (!$resultado)
        {
                exit("Error en la consulta");
           }
        else
        {
           echo "Comando Enviado";
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','desactivar
salida1','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else{echo "Error";}
?>
```

3.3.9. ACCIÓN ACTIVAR SALIDA2 (ACTIVAR_SALIDA2).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar=$_POST['activar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar=="activar")
{
    $db = odbc_connect($base_de_datos,$usu,$pas);
    if (!$db)
    {
```

```
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
    }
    else
    {
        $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
        $valor = odbc_fetch_array($resultado);
        $id_equipo=$valor['id_equipo'];

        $consulta1=sprintf("UPDATE estados SET comando='%s21' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
        if (!$resultado)
        {
                exit("Error en la consulta");
            }
        else
        {
            echo "Comando Enviado";
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar
salida2','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else
{
    echo "Error";
}
?>
```

3.3.10. ACCIÓN DESACTIVAR SALIDA2 (DESACTIVAR_SALIDA2.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$desactivar=$_POST['desactivar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($desactivar=="desactivar")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s20' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
      else
      {
         echo "Comando Enviado";
      }
      date_default_timezone_set('America/La_Paz');
      $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','desactivar
salida2','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
      odbc_exec($db,$consulta1);
      odbc_free_result($resultado);
      odbc_close($db);
   }
   require_once 'Request.php';
   $request = new Request();
```

```
   $request->addParam("cliente",$cliente);
   $request->addParam("codigo_cliente",$codigo_cliente);
   $request->addParam("nick",$nick);
   $request->addParam("salir",0);
   $request->forward("monitorear.php");
}
else
{
   echo "Error";
}
?>
```

3.3.11. ACCIÓN ACTIVAR SALIDA3 (ACTIVAR_SALIDA3.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar=$_POST['activar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar=="activar")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
         print "<p>Imposible conectarse con la base de datos    /p>";
         exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s31' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      //echo $consulta1;
      //exit();
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
```

```
        else
        {
           echo "Comando Enviado";
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar
salida3','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else
{
    echo "Error";
}
?>
```

3.3.12. ACCIÓN DESACTIVAR SALIDA3 (DESACTIVAR_SALIDA3.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$desactivar=$_POST['desactivar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($desactivar=="desactivar")
{
    $db = odbc_connect($base_de_datos,$usu,$pas);
    if (!$db)
    {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
    }
```

```
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s30' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
      else
      {
         echo "Comando Enviado";
      }
      date_default_timezone_set('America/La_Paz');
      $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','desactivar
salida3','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
      //echo $consulta1;
      //exit();
      odbc_exec($db,$consulta1);
      odbc_free_result($resultado);
      odbc_close($db);
   }
   require_once 'Request.php';
   $request = new Request();
   $request->addParam("cliente",$cliente);
   $request->addParam("codigo_cliente",$codigo_cliente);
   $request->addParam("nick",$nick);
   $request->addParam("salir",0);
   $request->forward("monitorear.php");
}
else
{
   echo "Error";
}
?>
```

3.3.13. ACCIÓN ACTIVAR ALARMA (ACTIVAR_ALARMA.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar=$_POST['activar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar=="activar")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s41' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      //echo $consulta1;
      //exit();
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
         }
      else
      {
         echo "Comando Enviado";
      }
      date_default_timezone_set('America/La_Paz');
      $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar
alarma','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
      odbc_exec($db,$consulta1);
      odbc_free_result($resultado);
      odbc_close($db);
   }
```

```
   require_once 'Request.php';
   $request = new Request();
   $request->addParam("cliente",$cliente);
   $request->addParam("codigo_cliente",$codigo_cliente);
   $request->addParam("nick",$nick);
   $request->addParam("salir",0);
   $request->forward("monitorear.php");
}
else
{
   echo "Error";
}
?>
```

3.3.14. ACCIÓN ACTIVAR MODO PRESENTE (ACTIVAR_ALARMA_MP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$activar_mp=$_POST['activar_mp'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($activar_mp=="activar_mp")
{
   $db = odbc_connect($base_de_datos,$usu,$pas);
   if (!$db)
   {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
   }
   else
   {
      $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      $valor = odbc_fetch_array($resultado);
      $id_equipo=$valor['id_equipo'];

      $consulta1=sprintf("UPDATE estados SET comando='%s42' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
      $resultado=odbc_exec($db,$consulta1);
      if (!$resultado)
      {
            exit("Error en la consulta");
      }
```

```
        else
        {
           echo "Comando Enviado";
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','activar alarma
mp','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else
{echo "Error";}
?>
```

3.3.15. ACCIÓN DESACTIVAR ALARMA (DESACTIVAR_ALARMA.PHP).

```
<?php
$cliente=$_POST['cliente'];
$codigo_cliente=$_POST['codigo_cliente'];
$nick=$_POST['nick'];
$desactivar=$_POST['desactivar'];
$base_de_datos='clientes1';
$usu='';
$pas='';
if($desactivar=="desactivar")
{
    $db = odbc_connect($base_de_datos,$usu,$pas);
    if (!$db)
    {
        print "<p>Imposible conectarse con la base de datos    /p>";
        exit();
    }
    else
    {
        $consulta1=sprintf("SELECT id_equipo FROM estados WHERE
codigo_cliente='%s'",$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
```

```
        $valor = odbc_fetch_array($resultado);
        $id_equipo=$valor['id_equipo'];
        $consulta1=sprintf("UPDATE estados SET comando='%s40' WHERE
codigo_cliente='%s'",$id_equipo,$codigo_cliente);
        $resultado=odbc_exec($db,$consulta1);
        if (!$resultado)
        {
                exit("Error en la consulta");
        }
        else
        {
           echo "Comando Enviado";
        }
        date_default_timezone_set('America/La_Paz');
        $consulta1=sprintf("INSERT INTO historial
(cliente,codigo_cliente,nick,accion,fecha_accion) VALUES('%s','%s','%s','desactivar
alarma','%s')",$cliente,$codigo_cliente,$nick,date("d-m-Y H:i:s"));
        odbc_exec($db,$consulta1);
        odbc_free_result($resultado);
        odbc_close($db);
    }
    require_once 'Request.php';
    $request = new Request();
    $request->addParam("cliente",$cliente);
    $request->addParam("codigo_cliente",$codigo_cliente);
    $request->addParam("nick",$nick);
    $request->addParam("salir",0);
    $request->forward("monitorear.php");
}
else
{
    echo "Error";
}
?>
```

3.4. TRAMAS DE DATOS USADOS EN LA COMUNICACIÓN CLIENTE SERVIDOR.

Las tramas de datos que se usaran en la comunicación ente el servidor y el equipo de rastreo serán de dos tipos.

3.4.1. TRAMA DE DATOS DEL EQUIPO DE MONITOREO HACIA EL SERVIDOR.

La trama de datos tendrá la siguiente forma:

Smduxxxazzzzzzzzeeeeeeeessss

Donde.

smd, es el acrónimo del nombre del equipo de monitoreo (sistema de monitoreo digital).

uxxxx, es el identificador del equipo, xxxx puede variar de 0000 a 9999.

a, es el estado de la equipo de monitoreo y puede tener tres valores 0,1,2 (desactivado, activado y activado modo presente).

zzzzzzzz, es el estado de las zonas (entradas) del equipo de monitoreo, cada ***z*** representa a una zona en particular desde la ***zona0 hasta la zona7*** y pueden tener cuarto valores 0, 1, 2 y 3 (abierto, cerrado, cortocircuito y deshabilitado).

eeeeeeee, señaliza los eventos que se disparan en las zonas desde la ***zona0 hasta la zona7*** y pueden tener dos valores 0 y !=0 (evento no disparado, evento disparado).

ssss, es el estado de las salidas desde la salida0 hasta salida3 y puede tener dos valores 0 y 1 (salida desactivada, salida activada).

3.4.2. TRAMA DE DATOS DEL SERVIDOR AL EQUIPO DE MONITOREO.

Esta trama de datos tendrá la siguiente forma.

smduxxxxse

smduxxxx, lo mismo que el anterior caso.

s, es la salida que se desea cambiar y puede ser de 0 a 4

e, es el estado al que se desea cambiar y puede ser 0 o 1.

Cuando ***s es 4*** se refiere al estado del equipo de monitoreo y en ese caso ***e*** puede tener los siguientes valores 0, 1 y 2 (desactivado, activado y activado modo presente).

3.5. SERVIDOR SOCKETS.

Este programa es el encargado de escuchar y atender a lo sockets clientes (equipos de los usuarios).

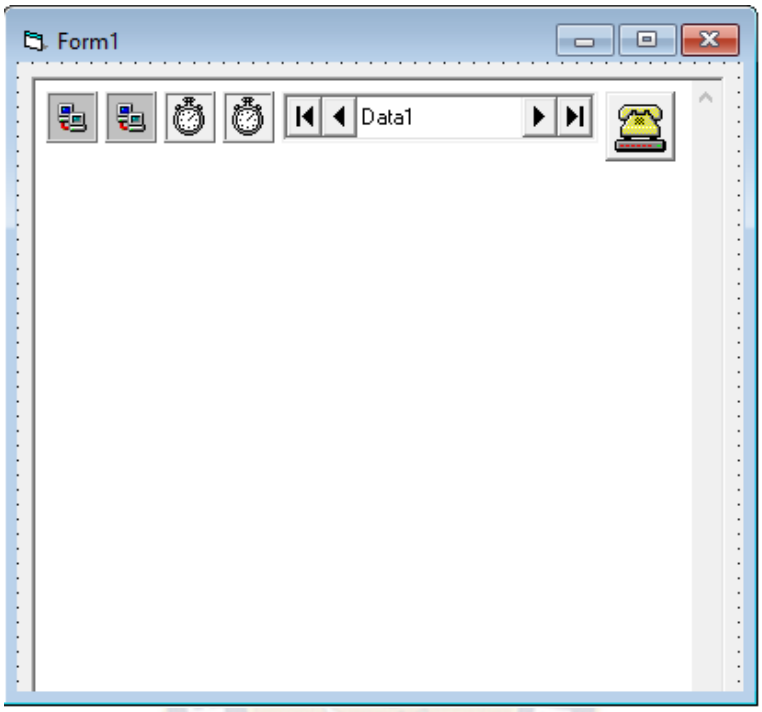


Figura 0.6 Formulario del servidor de sockets.

Variables globales.

```
'*************************************************************************
'Variables para el manejo de la base de datos
'*************************************************************************
Dim Wrk As Workspace        'creo un espacio de trabajo.
Dim MIWRK As Database       'creo una base de datos.
Dim MIrec As Recordset      'creo una tabla temporal.
Dim lstConect As String     'definición de una variable localstringconect.
Dim MIsql As String         'creo una variable texto para contener sql.
'*************************************************************************
Dim bandera_sms As Integer 'variable auxiliar que señaliza el envío de un sms.
Dim error As Integer        'variable señaliza un error.
'*************************************************************************
```

Evento load.

Pone en escucha al socket1, abre el puerto serie.

```
Private Sub Form_Load()
Form1.Caption = "http://" & Winsock1.LocalIP & ":" & Winsock1.LocalPort
escuchar_socket1                    'Escuchar clientes
MSComm1.PortOpen = True
```

```
bandera_sms = 0
End Sub
```

Evento unload.
Actualiza la base de datos a un estado que señaliza que todos los clientes están desconectados.

```
Private Sub Form_Unload(Cancel As Integer)
abrir_db
consulta ("UPDATE  estados SET numero_socket=-1")
cerrar_db
End Sub
```

Evento timer1.
Este evento se genera cada 200[ms] y está encargado junto con el puerto serie de leer los SMS que llegan a través del modem GSM cuando el equipo del cliente no está conectado al servidor.

```
Private Sub Timer1_Timer()
Static buffer As String
Static index_sms As String
Static i As Integer
Dim rx As String
While MSComm1.InBufferCount > 0
    rx = MSComm1.Input
    buffer = buffer & rx
    If rx = Chr(10) Then
      Text1.Text = Text1.Text & "**************************************" & vbCrLf
      Text1.Text = Text1.Text & buffer
      Text1.SelStart = Len(Text1.Text)
      Select Case i
         Case 0
            If InStr(buffer, "+CMTI:") > 0 Then
               badera_sms = 1
               index_sms = Mid(buffer, InStr(buffer, ",") + 1, 1)
               MSComm1.Output = "at+cmgr=" & index_sms & Chr(13)
               i = i + 1
            End If
         Case 1
            If StrComp(buffer, "OK" & vbCrLf) <> 0 Then
               verificar_datos buffer, -1
            Else
               MSComm1.Output = "at+cmgd=" & index_sms & Chr(13)
               i = 2
            End If
         Case 2
            If StrComp(buffer, "OK" & vbCrLf) <> 0 Then
```

```
            i = 0
            badera_sms = 0
          End If
      End Select
      buffer = ""
   End If
Wend
End Sub
```

Evento timer2.

Se ejecuta cada 200[ms] y está encargado de leer la base de datos y enviar los comandos a los equipos de los clientes conectados al servidor, algún equipo no estaría conectado entonces se enviara el comando vía SMS a través del modem GSM conectado al puerto serie.

```
Private Sub Timer2_Timer()
Dim rx As String
Dim buffer As String
Dim bandera As Integer
abrir_db
   If error = 1 Then
      error = 0
      Exit Sub
   End If

consulta "SELECT numero_socket,numero_equipo,id_equipo,comando,id_equipo FROM
estados"
MIrec.MoveFirst
Do Until MIrec.EOF                    'Mientras no llegue al final del registro
   If MIrec("comando") <> "-" Then
      bandera = 1
      Text1.Text = Text1.Text &
"**************************************************************" & vbCrLf
      Text1.SelStart = Len(Text1.Text)
      If StrComp(MIrec("numero_socket"), "-1") = 0 Then
         If badera_sms = 0 Then
            Timer1.Enabled = False
            MSComm1.Output = "at+cmgs=" & Chr(34) & MIrec("numero_equipo") & Chr(34)
& Chr(13)
            Do
               If MSComm1.InBufferCount > 0 Then
                  rx = MSComm1.Input
                  buffer = buffer & rx
               End If
               DoEvents
            Loop While InStr(buffer, ">") = 0
```

```
            Text1.Text = Text1.Text & buffer & vbCrLf
            Text1.SelStart = Len(Text1.Text)
            buffer = ""
            MSComm1.Output = MIrec("comando") & Chr(26)
            Do
               If MSComm1.InBufferCount > 0 Then
                  rx = MSComm1.Input
                  buffer = buffer & rx
               End If
               DoEvents
            Loop While InStr(buffer, "OK" & vbCrLf) = 0
            Text1.Text = Text1.Text & buffer & vbCrLf
            Text1.SelStart = Len(Text1.Text)
            buffer = ""
            Timer1.Enabled = True
         End If
      Else
         Winsock2(MIrec("numero_socket")).SendData (MIrec("comando") & vbCrLf)
         Text1.Text = Text1.Text & "Servidor: Socket" & MIrec("numero_socket") & " ---> " &
MIrec("comando") & vbCrLf
         Text1.SelStart = Len(Text1.Text)
      End If
   End If
   MIrec.MoveNext                    'Apunta al siguiente registro
Loop
If bandera = 1 Then
   consulta ("UPDATE estados SET comando ='-' WHERE comando <> '-'")
End If
cerrar_db
End Sub
```

Winsock1_ConnectionRequest.

Se ejecuta cada vez que un equipo cliente desea conectarse al servidor, atiende la conexión y destina los recursos para mantener al equipo cliente conectado al servidor.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
Dim numSocket As Integer
Text1.Text = Text1.Text & "**********************************" & vbCrLf
Text1.Text = Text1.Text & "Petición número " & requestID & vbCrLf
numSocket = NuevoSocket()                                    'creamos un nuevo socket
Winsock2(numSocket).Accept requestID          'aceptamos la conexión con el nuevo socket
Text1.Text = Text1.Text & "Sock" & numSocket & ": conexión aceptada, listo para interactuar."
& vbCrLf
Text1.SelStart = Len(Text1.Text)
End Sub
```

Evento Winsock2_Close.

Se ejecuta cada vez que el equipo cliente pierde conexión con el servidor y está encargado de actualizar la base de datos señalizando que el equipo cliente no está conectado al servidor.

```
Private Sub Winsock2_Close(index As Integer)
Winsock2(index).Close                    'cierra  la conexión
Text1.Text = Text1.Text &
"**********************************************************" & vbCrLf
Text1.Text = Text1.Text & "Sock" & index & ":Conexión cerrada por el Cliente." & vbCrLf
Text1.SelStart = Len(Text1.Text)          'desplegamos un mensaje en la ventana
abrir_db
consulta ("UPDATE estados SET numero_socket='-1' WHERE numero_socket=" & "'" & index
& "'")
cerrar_db
End Sub
```

Evento Winsock2_DataArrival.

Se ejecuta cada vez que el servidor recibe información del equipo cliente y actualiza la base de datos.

```
Private Sub Winsock2_DataArrival(index As Integer, ByVal bytesTotal As Long)
Dim buffer As String                     'variable para guardar los datos
Winsock2(index).GetData buffer, vbString  'obtenemos los datos y los guardamos en una
                                          'variable
Text1.Text = Text1.Text &
"**************************************************************" & vbCrLf
Text1.Text = Text1.Text & "Cliente: Socket" & index & " <-- " & buffer & vbCrLf
Text1.SelStart = Len(Text1.Text)
verificar_datos buffer, index
End Sub
```

Subrutinas para el manejo de los sockets.

```
Private Sub escuchar_socket1()
Winsock1.Close                   'cerramos cualquier conexión previa
Winsock1.Listen                  'deja el socket escuchando conexiones
End Sub

Private Function NuevoSocket() As Integer 'Carga un nuevo socket al arreglo y devuelve su
indice
Dim numElementos As Integer              'numero de sockets
Dim i As Integer                 'contador
```

```
numElementos = Winsock2.UBound          'obtiene la cantidad de Winsocks que tenemos
For i = 0 To numElementos               'recorre el arreglo de sockets
   If Winsock2(i).State = sckClosed Then 'si algún socket ya creado está inactivo
                                        'utiliza este mismo para la nueva conexión
      NuevoSocket = i                   'retorna el índice
      Exit Function                     'abandona la función
   End If
Next
                                        'si no encuentra sockets inactivos
                                        'crea uno nuevo y devuelve su identidad
Load Winsock2(numElementos + 1)         'carga un nuevo socket al arreglo
NuevoSocket = Winsock2.UBound           'devuelve el nuevo índice
End Function
```

Subrutinas para el manejo de la base de datos.

```
Private Sub abrir_db()
On Error GoTo error_db:
Set Wrk = CreateWorkspace("clientes", "", "", dbUseODBC)
lstConect = "ODBC;UID=;PWD=;eventos"
Set MIWRK = Wrk.OpenDatabase("clientes1", dbDriverNoPrompt, False, lstConect)
error_db: If Err.Number <> 0 Then
          error = 1
        End If
End Sub

Private Sub cerrar_db()
MIrec.Close
MIWRK.Close
Wrk.Close
End Sub

Private Sub consulta(MIsql)
Set MIrec = MIWRK.OpenRecordset(MIsql, dbOpenSnapshot)
End Sub
```

Subrutinas y funciones.

```
Private Sub verificar_datos(buffer As String, index As Integer)
Const identificador = "smdu"          'identificador de id del equipo
Const n_id = 4                        'numero de caracteres del id del equipo
Const n_datos = 21          'numero de caracteres de la trama de datos que el equipo reporta
Dim id_equipo As String
Dim datos As String
If Mid(buffer, 1, Len(identificador)) = identificador Then
   id_equipo = Mid(buffer, 1, Len(identificador) + n_id)
```

```
   datos = Mid(buffer, Len(identificador) + n_id + 1, n_datos)
   actualizar_estados id_equipo, datos, index
End If
End Sub

Private Sub actualizar_estados(id_equipo As String, estados As String, numero_socket As
Integer)
Dim n As Single
abrir_db
   If error = 1 Then
      error = 0
      Exit Sub
   End If
consulta "UPDATE estados SET numero_socket=" & "'" & numero_socket & "'" & " WHERE
id_equipo=" & "'" & id_equipo & "'"
If Mid(estados, 1, 1) = 0 Then
   consulta ("UPDATE estados SET alarma='desactivada' WHERE id_equipo=" & "'" &
id_equipo & "'")
ElseIf Mid(estados, 1, 1) = 1 Then
   consulta ("UPDATE estados SET alarma='activada' WHERE id_equipo=" & "'" & id_equipo
& "'")
Else
   consulta ("UPDATE estados SET alarma='modo_presencia' WHERE id_equipo=" & "'" &
id_equipo & "'")
End If
For n = 0 To 7 Step 1
   If Mid(estados, n + 2, 1) = 0 Then
      consulta ("UPDATE estados SET zona" & n & "='abierto' WHERE id_equipo=" & "'" &
id_equipo & "'")
   ElseIf Mid(estados, n + 2, 1) = 1 Then
      consulta ("UPDATE estados SET zona" & n & "='cerrado' WHERE id_equipo=" & "'" &
id_equipo & "'")
   ElseIf Mid(estados, n + 2, 1) = 2 Then
      consulta ("UPDATE estados SET zona" & n & "='corto' WHERE id_equipo=" & "'" &
id_equipo & "'")
   Else
      consulta ("UPDATE estados SET zona" & n & "='N' WHERE id_equipo=" & "'" &
id_equipo & "'")
   End If
Next
For n = 0 To 7 Step 1
   If Mid(estados, n + 10, 1) = 0 Then
      consulta ("UPDATE estados SET evento_zona" & n & "='no_disparado' WHERE
id_equipo=" & "'" & id_equipo & "'")
   Else
      consulta ("UPDATE estados SET evento_zona" & n & "='disparado' WHERE id_equipo="
& "'" & id_equipo & "'")
```

```
   End If
Next
For n = 0 To 3 Step 1
   If Mid(estados, n + 18, 1) = 0 Then
      consulta ("UPDATE estados SET salida" & n & "='desactivado' WHERE id_equipo=" & """ & id_equipo & """)
   Else
      consulta ("UPDATE estados SET salida" & n & "='activado' WHERE id_equipo=" & """ & id_equipo & """)
   End If
Next
cerrar_db
End Sub
```

3.6. HARDWARE DEL EQUIPO DE MONITOREO.

3.6.1. HARDWARE DE TECLADO Y LCD.

Este hardware está encargado de mostrar al cliente el estado del equipo de monitoreo, en el LCD aparecerá la información del estado de la alarma, estado de las zonas y el estado de las salidas.

El teclado proporcionara al cliente el interface para introducir su código y pode así activar la alarma en sus tres estados (activado, desactivado y activado modo presente).

Regulador de voltaje.

Datos.
Vi=14.5[v] Vo=5[v] Io=500[mA] Fs=100K[Hz]
Se utilizara el convertidor DC-DC mc34063 y las ecuaciones del punto 11.14.9

$$\delta = \frac{Vo}{Vi} \quad \delta = \frac{5}{14.5} \quad \delta = 0.345$$

$$Lmin = \frac{(Vi - Vo) * \delta}{2 * Iomin * Fs} \quad Lmin = \frac{(14.5 - 5) * 0.345}{2 * 0.5 * 100k} \quad Lmin = 33u[H]$$

Como el valor mínimo del inductor es de 33u[H] se consiguió un inductor de **2.7m[H]** con este nuevo valor de inductancia se podrá calcular las corrientes pico que atraviesan al inductor.

$$I2 - I1 = \frac{(Vi - Vo) * \delta}{Lmin * Fs} \quad I2 - I1 = \frac{(14.5 - 5) * 0.345}{2.7m * 100K} \quad I2 - I1 = 0.0121$$

De la corriente promedio de salida.

$$\frac{(I2+I1)}{2} = Io \quad I2 + I1 = 2 * Io \quad I2 + I1 = 2 * 0.5 = 1$$

Resolviendo el sistema de ecuaciones se tiene.

$$\boldsymbol{I2 = 0.506[A]} \quad I1 = 0.494[A]$$

I2 es la corriente pico máxima que circulara por el inductor por tanto es la misma corriente pico que soportara el transistor y el diodo. El diodo que se usara es el **1N4819** que es de alta velocidad t de 1[A].

El capacitor de salda con un rizado de salida de 0.01[v].

$$C = \frac{Iomax * \delta}{Fs * VoRPP} \quad C = \frac{0.5 * 0.345}{100K * 0.01} \quad C = 172u[F] \rightarrow \boldsymbol{C = 220u[F]}$$

Capacitor de CT para el oscilador de mc34063.

$$CT = 4x10^{-5} * \frac{\delta}{Fs} \quad CT = 4x10^{-5} * 3.45u \quad CT = 138p[F] \rightarrow \boldsymbol{CT = 270p[F]}$$

Los resistores de realimentación.

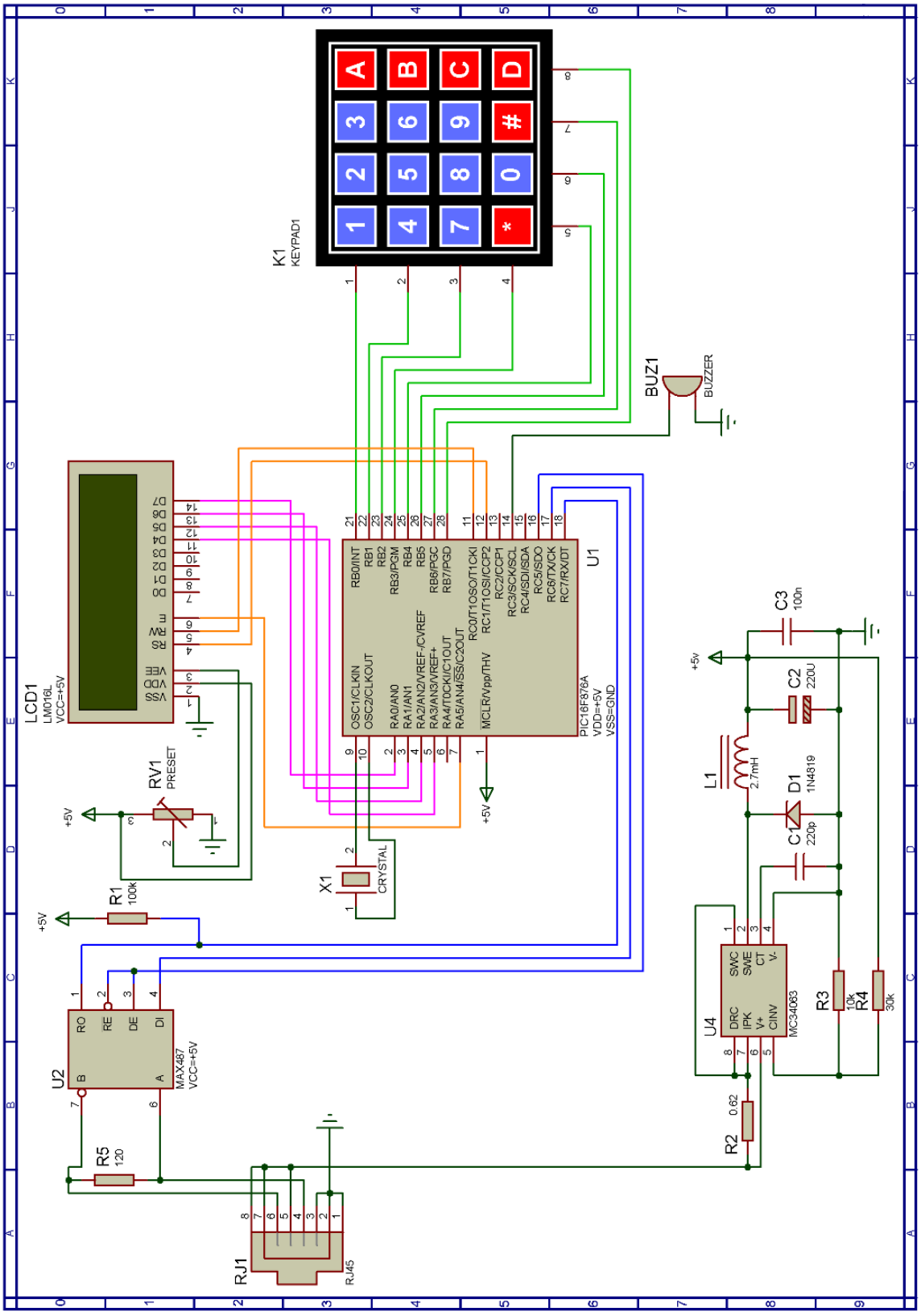
$$Vo = 1.25 * \left(1 + \frac{R2}{R1}\right) \quad si\ \boldsymbol{R2 = 30K} \quad Vo = 1.25\left(1 + \frac{30k}{R1}\right) \quad \boldsymbol{R1 = 10K}$$

Resistor limitador de corriente.

$$RSC = \frac{0.3}{Imax} \quad RSC = \frac{0.3}{0.5} \quad RSC = 0.6 \rightarrow \boldsymbol{RSC = 0.62[\Omega]}$$

Para el envío de la alimentación y datos se utiliza cable UTP categoría 5, los datos se transmitirán de modo asíncrono RS232 mediante el bus RS458.

Esquema del circuito.



3.6.2. HARDWARE DEL EQUIPO DE MONITOREO.

El equipo de monitoreo también contara con el conversor DC-DC mc34063 y se mantienen los mismos parámetros que en el anterior caso a excepción del voltaje de salida que en este caso será de 4.4[v] debido a que el modem GSM/GPRS soporta un voltaje de alimentación máximo de 4.5[v] y el microcontrolador PIC18f4550 requiere mínimo 4.2[v] para trabajar a máxima velocidad.

Datos.
Vo=4.4[v]

$$Vo = 1.25 * \left(1 + \frac{R2}{R1}\right) \quad si\ \boldsymbol{R2 = 51K[\Omega] \rightarrow R1 = 10K[\Omega]}$$

El Equipo tendrá 8 entradas denominadas zonas0 a zona7, estas zonas podrán ser configuradas para ser habilitadas o deshabilitadas, en el caso de estar deshabilitadas el LCD mostrara "-" en el lugar correspondiente a la zona. En caso de estar la zona habilitada entonces se mostrara una "A" cuando la zona este abierta, una "C" cuando la zona este cerrada y "*" cuando la zona sea saboreada con un corto circuito.

A cada zona se le podrá configurar un evento que podrá ser disparado cuando la zona este abierta o cerrada, también se puede configurar si el evento se dispara cuando la alarma del equipo de monitoreo este activado, desactivado o modo presente así también se podrá añadir un tiempo de retardo de zona desde 0 que es evento inmediato hasta 255 segundos de retardo.

Las salidas podrán ser configuradas para activarse o desactivarse según la asignación de algún evento.

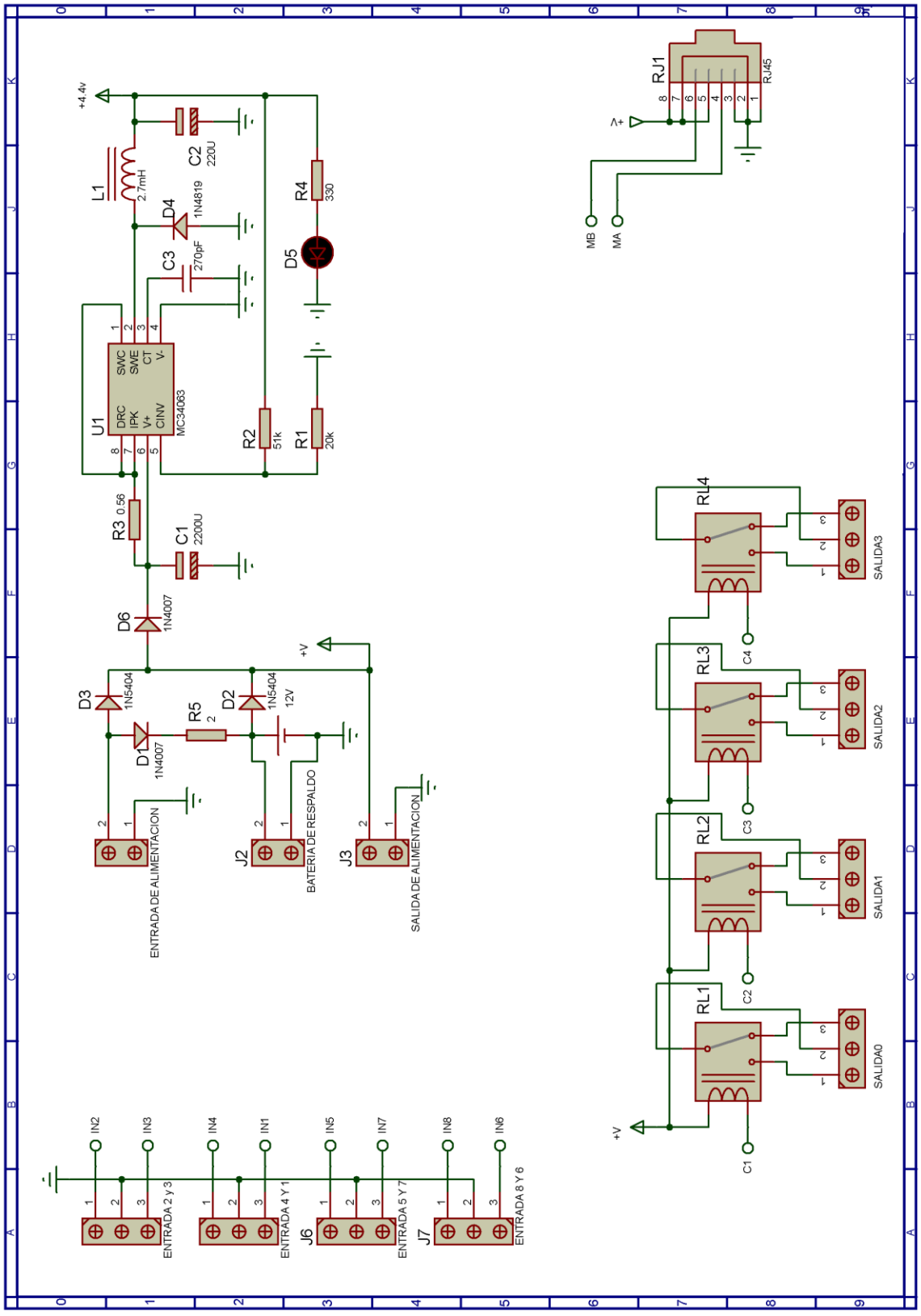
A una salida se le podrán asignar más de una zona con su respectivo evento.

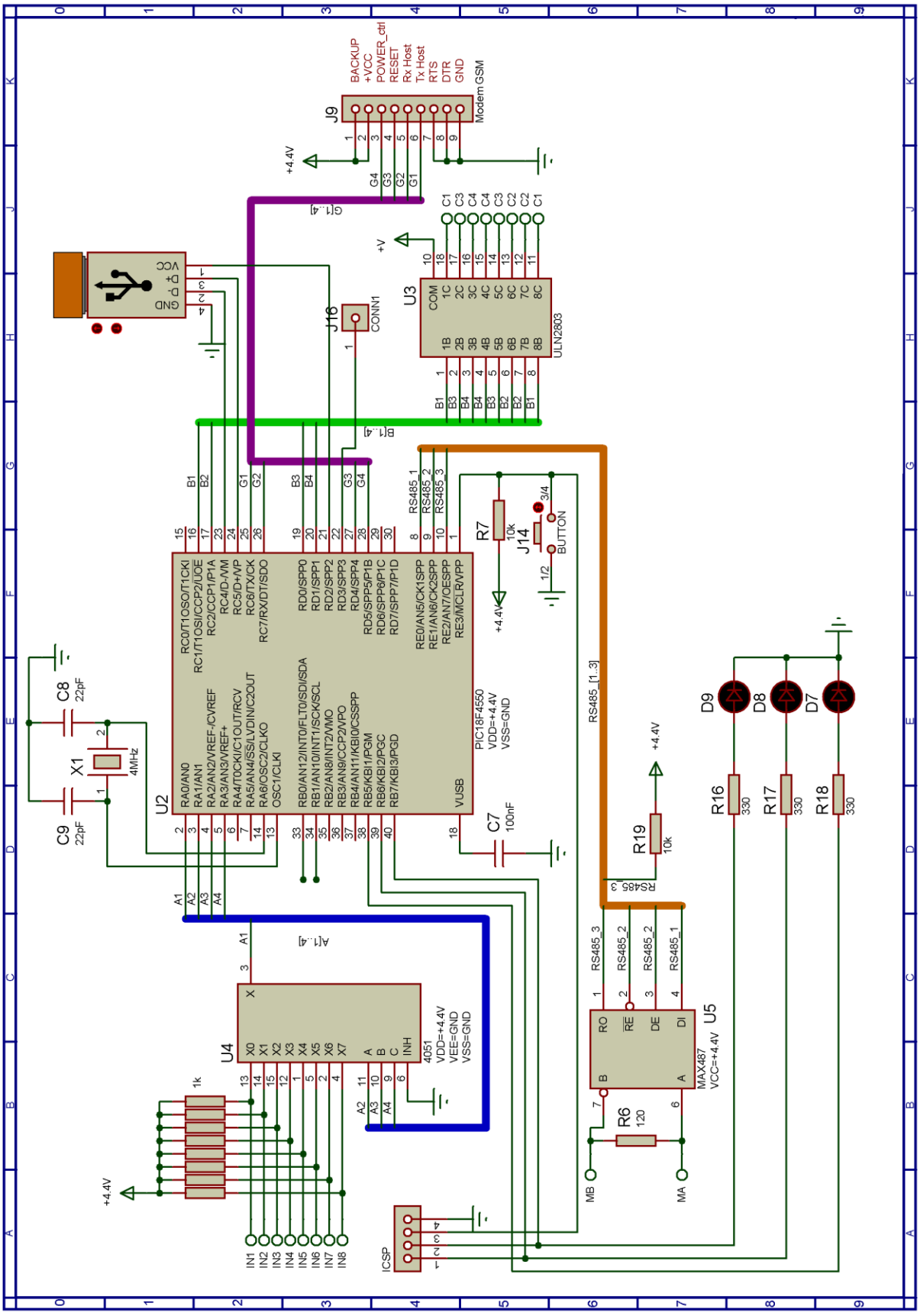
A cualquier zona se le podrá asigne el evento especial de "pánico", este evento se disparara sin importar si el la alarma del equipo este activada, desactivada o en modo presente.

A cada salida se le podrá programar un tiempo de activado desde 0 que es activado permanente hasta 255 segundos.

Cuando de desactive la alarma del equipo del monitoreo mediante la introducción el código se desactivaran todas las salidas.

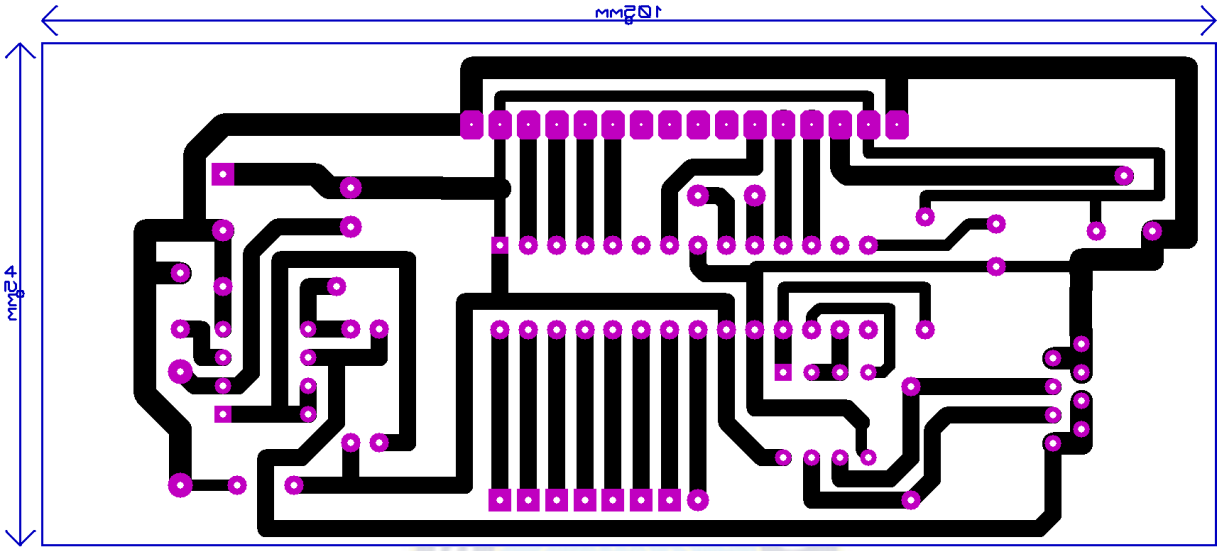
Esquena del equipo de monitoreo.



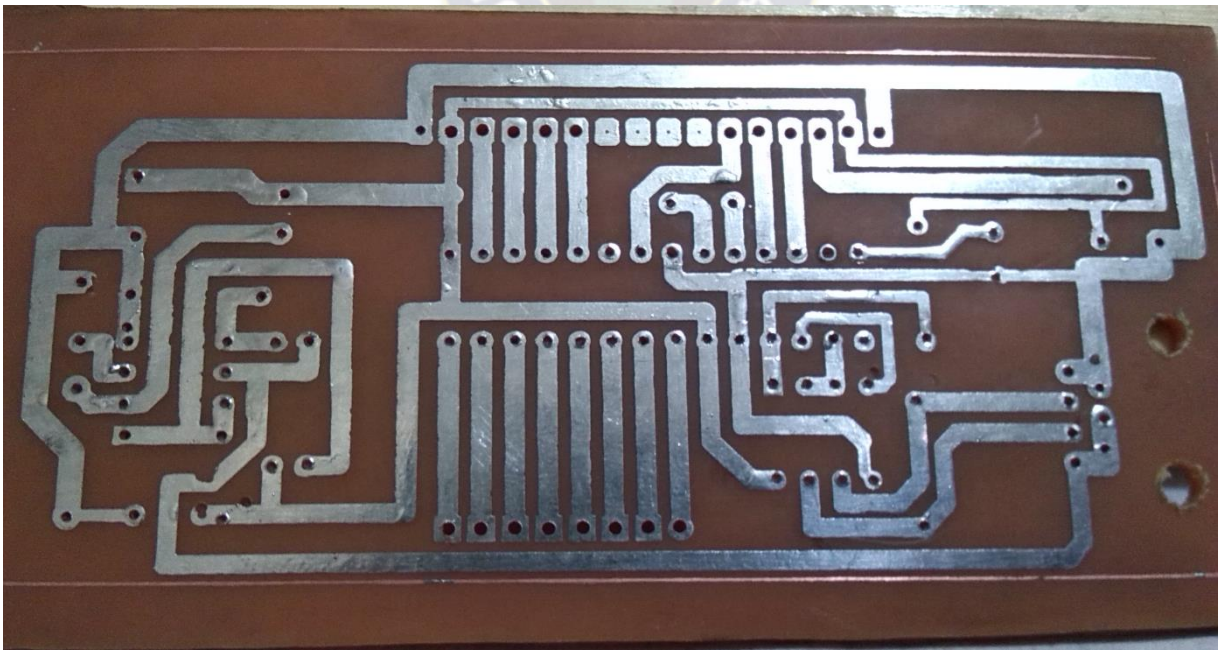

J9
BACKUP
+VCC
POWER_ctrl
RESET
Rx Host
Tx Host
RTS
DTR
GND
Modem GSM
+4.4V
U3
ULN2803
J16
CONN1
U2
PIC18F4550
VDD=+4.4V
VSS=GND
C8
22pF
C9
22pF
X1
4MHz
C7
100nF
R7
10k
J14
BUTTON
R19
10k
U4
4051
VDD=+4.4V
VEE=GND
VSS=GND
U5
MAX487
VCC=+4.4V
R6
120
R16
330
R17
330
R18
330
D9
D8
D7
ICSP
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
MB
MA

3.6.3. IMPLEMENTACIÓN DE LOS CIRCUITOS.

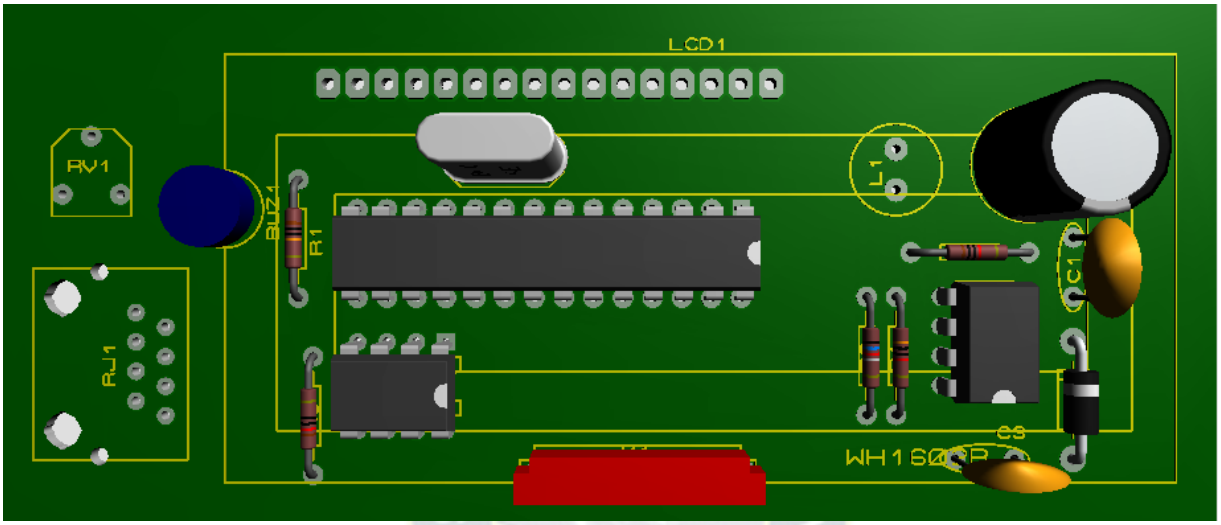
Diseño de la placa de circuito impreso del teclado.



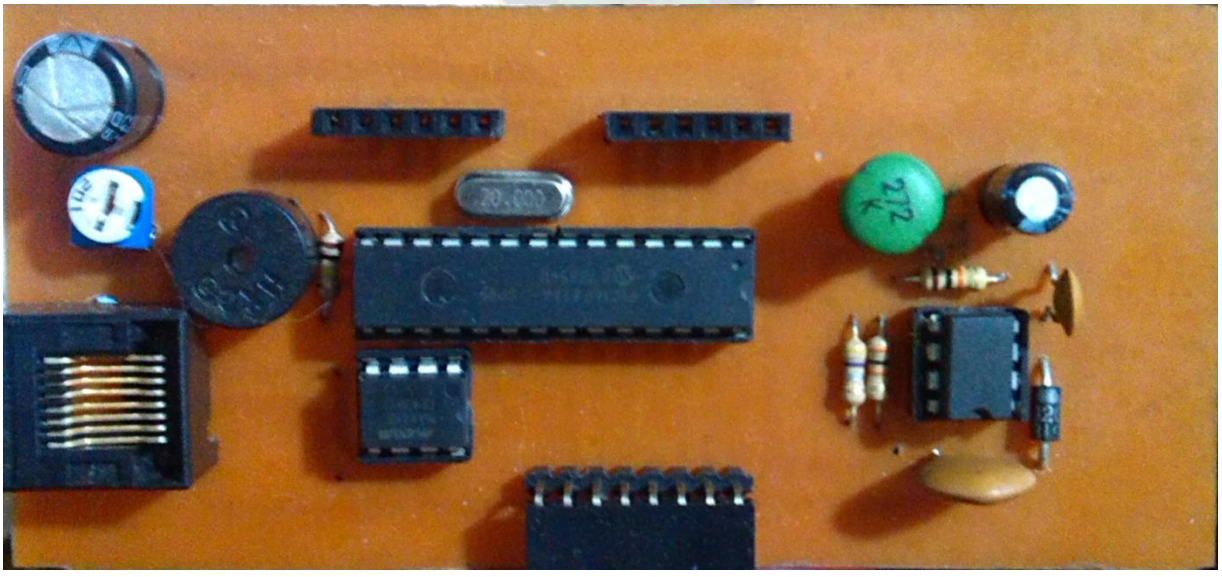
Placa de circuito impreso del teclado.



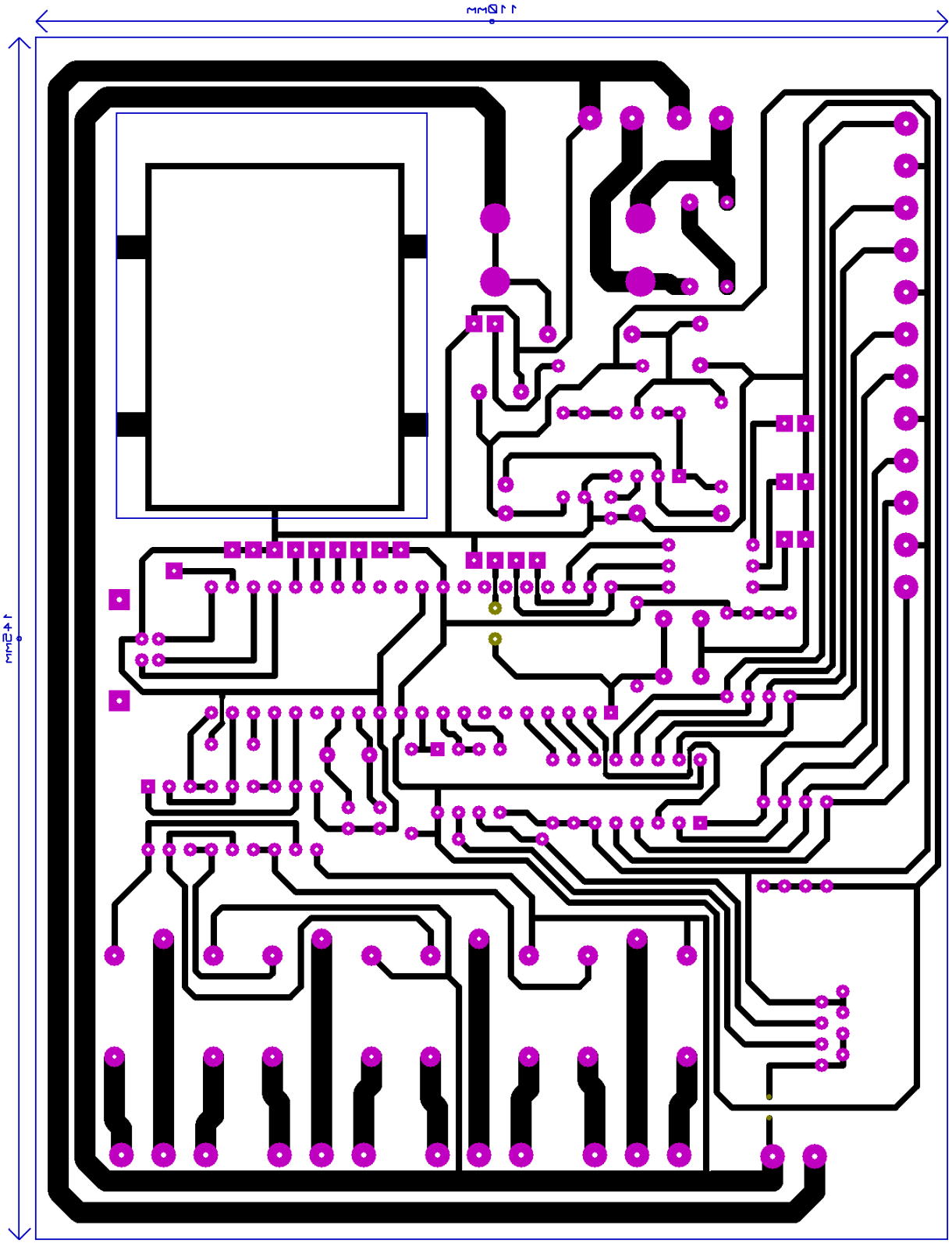
Modelo del circuito ya implementado.



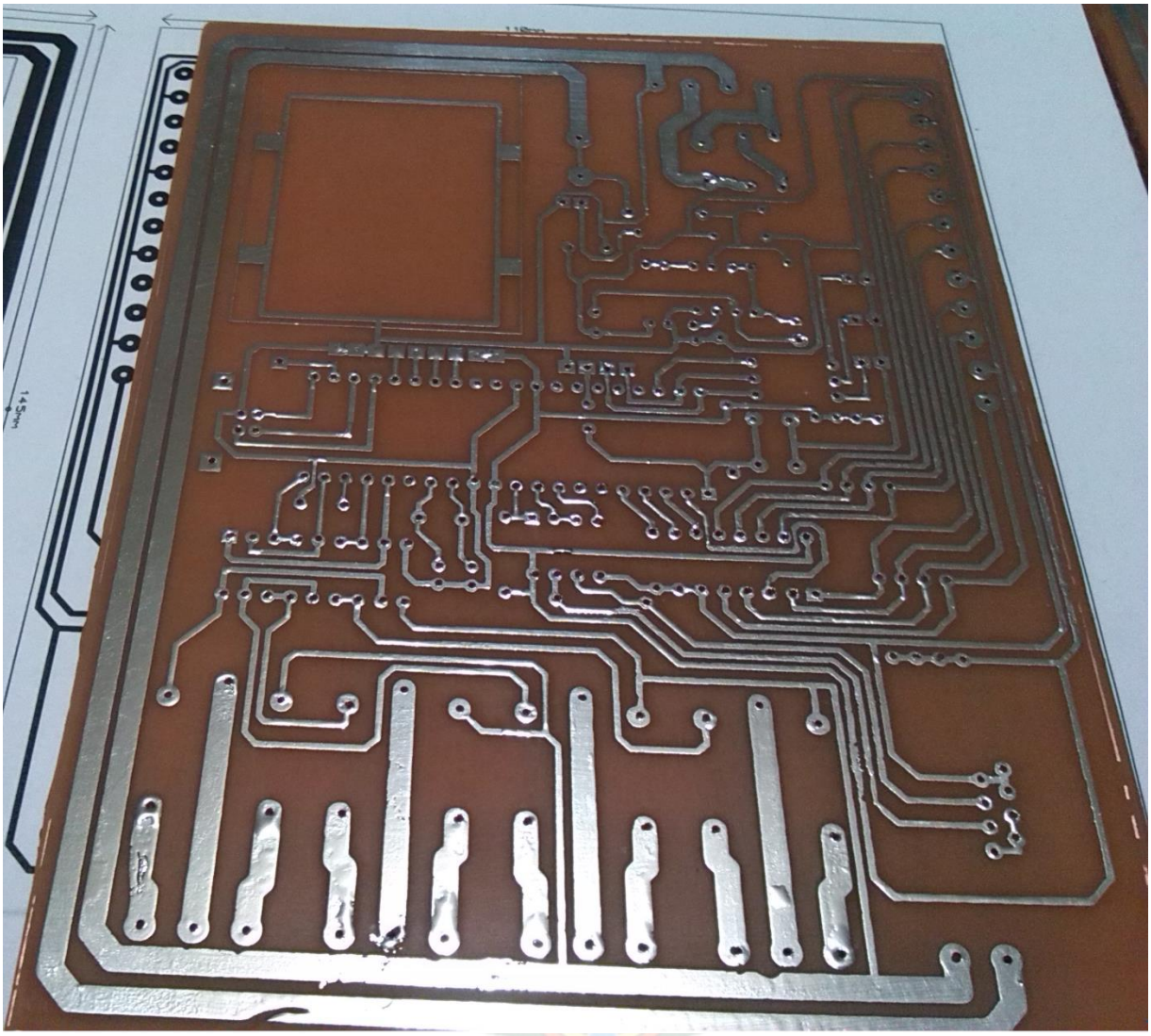
Circuito Implementado.



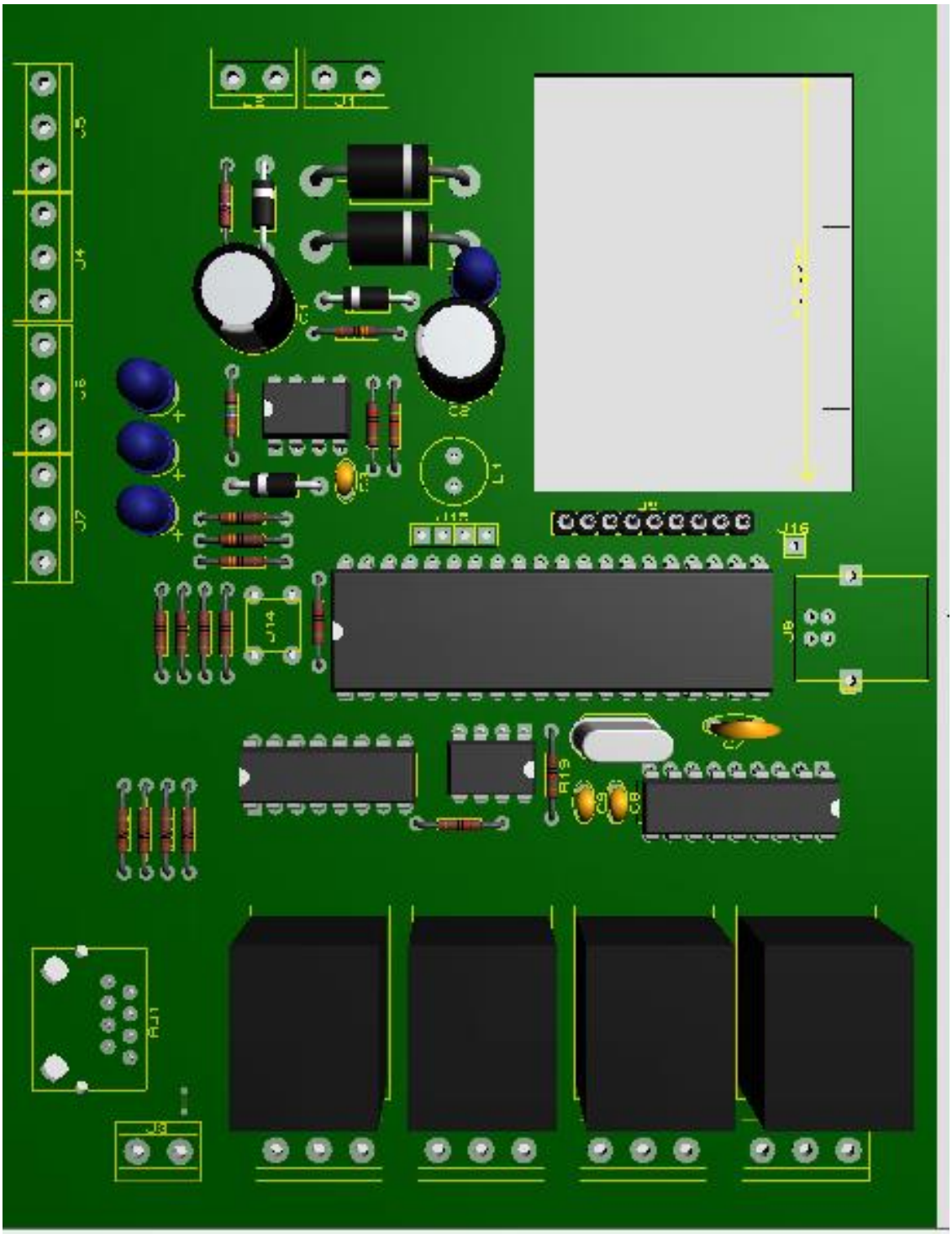
Diseño de la placa de circuito impreso del circuito de monitoreo.



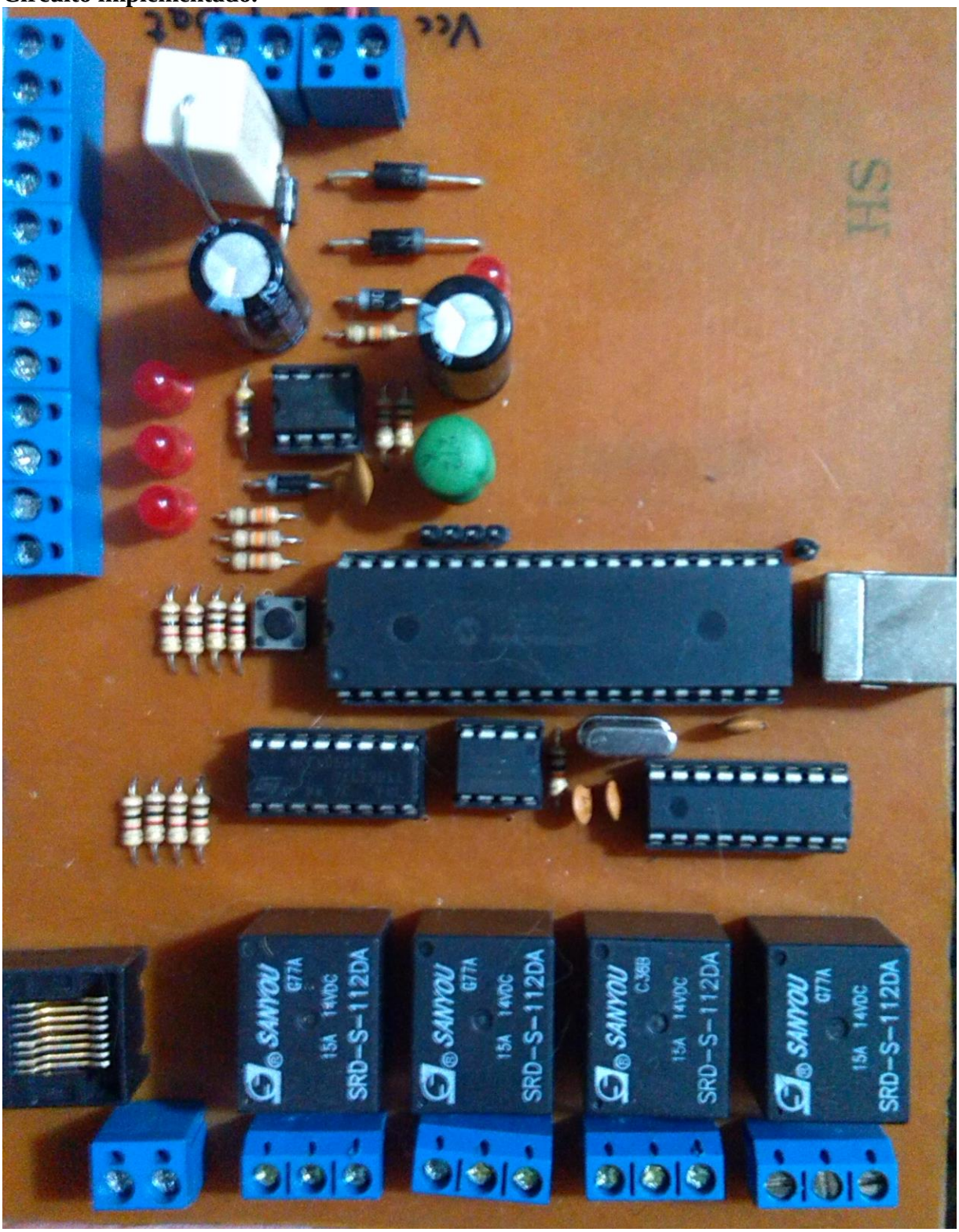
Placa de circuito impreso.



Modelo del circuito implementado.



Circuito implementado.



3.7. FIRMWARE DEL EQUIPO DE MONITOREO.

El firmware del equipo de monitoreo está dividido en dos partes, una la corresponde al teclado y lcd para que el cliente pueda introducir su código y así poder activar o desactivar la alarma, la otra que corresponde al que gestionara toda la información de las zonas, salidas y comunicación con el modem GSM/GPRS para el envío de los reportes.

El firmware está escrito en el compilador para CCS PicC versión 5.030.

3.7.1. FIRMWARE DEL TECLADO Y LCD.

Cabecera de inicialización del hardware, declaración de variables globales e inclusión de librerías.

teclado.h

```
/*****************************************************************************/
#include <16F876A.h>
#device ADC=10
#device PASS_STRINGS=IN_RAM
/*****************************************************************************/
#FUSES NOWDT            //No Watch Dog Timer
#FUSES PUT              //Power Up Timer
#FUSES BROWNOUT         //Reset when brownout detected
#FUSES NOLVP            //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD            //Data EEPROM Code Protected
#FUSES NOWRT            //Program memory not write protected
#FUSES NOPROTECT        //Code protected from reads
/*****************************************************************************/
#use delay(crystal=20MHz,restart_wdt)
/*****************************************************************************/
#use rs232(uart1,baud=115200,enable=pin_c5,restart_wdt,errors)
/*****************************************************************************/
#define LCD_ENABLE_PIN PIN_A5
#define LCD_RS_PIN PIN_C1
#define LCD_RW_PIN PIN_C0
#define LCD_DATA4 PIN_A3
#define LCD_DATA5 PIN_A2
#define LCD_DATA6 PIN_A1
#define LCD_DATA7 PIN_A0
#include <lcd.c>
/*****************************************************************************/
#define use_portb_kbd
#include <kbd_4x4.c>
/*****************************************************************************/
char buffer_rx[50];
char buffer_teclado[10];
```

```
char bandera_cadena_teclado_listo=0;
char bandera_datos=0;
unsigned int8 j=8;
/************************************************************************/
#define buzzer pin_c3
/************************************************************************/
```

Librería modificada para el manejo de teclado matricial 4x4.

```
Kbd_4x4.c
////////////////////////////////////////////////////////////////////////////////////////////////////
////                        KBD.C                                                               ////
////              Generic keypad scan driver                                                    ////
////                                                                                            ////
////  kbd_init()   Must be called before any other function.                                    ////
////  c = kbd_getc(c)  Will return a key value if pressed or /0 if not                          ////
////                This function should be called frequently so as                             ////
////                not to miss a key press.                                                    ////
////                                                                                            ////
////////////////////////////////////////////////////////////////////////////////////////////////////
////       (C) Copyright 1996,2003 Custom Computer Services                                     ////
//// This source code may only be used by licensed users of the CCS C                          ////
//// compiler.  This source code may only be distributed to other                              ////
//// licensed users of the CCS C compiler.  No other use, reproduction                         ////
//// or distribution is permitted without written permission.                                  ////
//// Derivative programs created using this software in object code                            ////
//// form are not restricted in any way.                                                       ////
////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////// The following defines the keypad layout on port D

// Un-comment the following define to use port B
// #define use_portb_kbd TRUE
// Make sure the port used has pull-up resistors (or the LCD) on
// the column pins
#if defined use_porta_kbd
  #byte kbd = getenv("SFR:PORTA")
#endif
#if defined use_portb_kbd
   #byte kbd = getenv("SFR:PORTB")
#endif
#if defined use_portc_kbd
   #byte kbd = getenv("SFR:PORTC")
#endif
#if defined use_portd_kbd
   #byte kbd = getenv("SFR:PORTD")
#endif

#if defined use_porta_kbd
```

```
  #define set_tris_kbd(x) set_tris_a(x)
#endif
#if defined use_portb_kbd
  #define set_tris_kbd(x) set_tris_b(x)
#endif
#if defined use_portc_kbd
  #define set_tris_kbd(x) set_tris_c(x)
#endif
#if defined use_portd_kbd
  #define set_tris_kbd(x) set_tris_d(x)
#endif
//Keypad connection:   (for example column 0 is B2)
//              Bx:
#ifdef blue_keypad  ////////////////////////////////// For the blue keypad
#define COL0 (1 << 2)
#define COL1 (1 << 3)
#define COL2 (1 << 6)
#define ROW0 (1 << 4)
#define ROW1 (1 << 7)
#define ROW2 (1 << 1)
#define ROW3 (1 << 5)
#else ////////////////////////////////////////////////// For the black keypad
//!#define COL0 (1 << 4)
//!#define COL1 (1 << 5)
//!#define COL2 (1 << 6)
//!#define COL3 (1 << 7)
//!#define ROW0 (1 << 0)
//!#define ROW1 (1 << 1)
//!#define ROW2 (1 << 2)
//!#define ROW3 (1 << 3)
#define COL0 (1 << 0)
#define COL1 (1 << 1)
#define COL2 (1 << 2)
#define COL3 (1 << 3)
#define ROW0 (1 << 4)
#define ROW1 (1 << 5)
#define ROW2 (1 << 6)
#define ROW3 (1 << 7)
#endif
#define ALL_ROWS (ROW0|ROW1|ROW2|ROW3)
#define ALL_PINS (ALL_ROWS|COL0|COL1|COL2|COL3)
// Keypad layout:
//!char const KEYS[4][4] = {{'1','2','3','A'},
//!                         {'4','5','6','B'},
//!                         {'7','8','9','C'},
//!                         {'*','0','#','D'}};
```

```
char const KEYS[4][4] =      {{'1','4','7','*'},
                             {'2','5','8','0'},
                             {'3','6','9','#'},
                             {'A','B','C','D'}};
#define KBD_DEBOUNCE_FACTOR 33    // Set this number to apx n/333 where
                          // n is the number of times you expect
                          // to call kbd_getc each second
//!#define KBD_DEBOUNCE_FACTOR 2
void kbd_init() {}
char kbd_getc( ) {
  static BYTE kbd_call_count;
  static int1 kbd_down;
  static char last_key;
  static BYTE col;
  BYTE kchar;
  BYTE row;
  kchar='\0';
  if(++kbd_call_count>KBD_DEBOUNCE_FACTOR) {
    switch (col) {
     case 0   : set_tris_kbd(ALL_PINS&~COL0);
                kbd=~COL0&ALL_PINS;
                break;
     case 1   : set_tris_kbd(ALL_PINS&~COL1);
                kbd=~COL1&ALL_PINS;
                break;
     case 2   : set_tris_kbd(ALL_PINS&~COL2);
                kbd=~COL2&ALL_PINS;
                break;
     case 3   : set_tris_kbd(ALL_PINS&~COL3);
                kbd=~COL3&ALL_PINS;
                break;
    }
    if(kbd_down) {
     if((kbd & (ALL_ROWS))==(ALL_ROWS)) {
       kbd_down=FALSE;
       kchar=last_key;
       last_key='\0';
     }
    } else {
      if((kbd & (ALL_ROWS))!=(ALL_ROWS)) {
        if((kbd & ROW0)==0)
          row=0;
        else if((kbd & ROW1)==0)
          row=1;
        else if((kbd & ROW2)==0)
          row=2;
        else if((kbd & ROW3)==0)
```

```
            row=3;
          last_key =KEYS[row][col];
          kbd_down = TRUE;
        } else {
          ++col;
          if(col==4)
            col=0;
        }
      }
    kbd_call_count=0;
  }
 set_tris_kbd(ALL_PINS);
 return(kchar);
}
```

Codigo fuente.

teclado.c

```
#include <teclado.h>
#int_timer1
void interrupcion_timer1()
{
  setup_timer_1(T1_DISABLED);
  output_low(buzzer);
}
#int_rda
void interrupcion_puerto_serie()
{
  gets(buffer_rx);
  if(buffer_rx[0]=='1')
  {
    if(bandera_cadena_teclado_listo==1)
    {
      bandera_cadena_teclado_listo=0;
      printf("%s\r",buffer_teclado);
    }
    else
    {
      printf("nan\r");
    }
  }
  else
  {
    bandera_datos=1;
  }
}
void leer_teclado()
```

```
{
  static unsigned int8 n;                  //contador de teclas presionadas
  char tecla;                                    //tecla presionada
  tecla=kbd_getc();
  if(tecla!=0)
  {
    switch(tecla)
    {
      case 'D':                                 //si tecla es igual a D
      {
        if(n!=0)                    //si hubo una tecla presionada antes
        {
          buffer_teclado[n]=0;            //almacena el fin de cadena de texto
          n=0;
          j=8;
          bandera_cadena_teclado_listo=1;
          lcd_putc("\f");
        }
        break;
      }
      case '*':                     //si tecla es igual a * (borrar datos)
      {
        lcd_gotoxy(8,2);              //cursor del lcd en fila 2 columna 8
        lcd_putc("        ");                  //borra la pantalla
        lcd_gotoxy(8,2);              //cursor del lcd en fila 2 columna 8
        n=0;
        break;
      }
      case '#':                                 //si tecla es igual a #
      {
        break;
      }
      default:                                       //en otros casos
      {
        if(n<sizeof(buffer_teclado)-1)     //contador de teclas menor a buffer-1
        {
          buffer_teclado[n++]=tecla;
          lcd_gotoxy(j++,2);
          lcd_putc('*');                //muestra en el lcd el símbolo *
        }
      }
    }
    output_high(buzzer);
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  }
}
void main()
```

```
{
  unsigned int8 i;
  port_b_pullups(true);                  //pull-up del puerto b activado
  kbd_init();                            //inicializa al teclado
  lcd_init(); //inicializa el lcd
  lcd_putc("reinicio");
  delay_ms(1000);

  lcd_putc("\flisto");
  setup_timer_1(T1_DISABLED);
  enable_interrupts(INT_TIMER1);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);
  while(true)
  {
    if(bandera_cadena_teclado_listo==0)
    {
      leer_teclado();
    }
    if(bandera_datos==1)
    {
      bandera_datos=0;
      switch(buffer_rx[0])
      {
        case '2':
        {
          lcd_putc("\f[Alarma]act:1\ndesac:2|pre:3|");
          j=15;
          break;
        }
        case '3':
        {
          switch(buffer_rx[1])
          {
            case 0:
            {
              lcd_putc("\a[D]");break;
            }
            case 1:
            {
              lcd_putc("\a[A]");break;
            }
            case 2:
            {
              lcd_putc("\a[P]");
            }
          }
```

```
            for(i=2;i<10;i++)
            {
              switch(buffer_rx[i])
              {
                case 0:
                {
                  lcd_putc("A");break;
                }
                case 1:
                {
                  lcd_putc("C");break;
                }
                case 2:
                {
                  lcd_putc("*");break;
                }
                case 3:
                {
                  lcd_putc("-");
                }
              }
            }
            lcd_putc("|");
            for(i=10;i<14;i++)
            {
              switch(buffer_rx[i])
              {
                case 0:
                {
                  lcd_putc("D");break;
                }
                case 1:
                {
                  lcd_putc("A");
                }
              }
            }
            lcd_putc("\nCodigo:");
            break;
          }
          case '4':{lcd_putc("\fError:");j=1;
          }
        }
      }
    }
  }
```

3.7.2. FIRMWARE DEL EQUIPO DE MONITOREO.

Cabecera de inicialización del hardware, declaración de variables, constantes e inclusión de librerías.

control.h

```
/*****************************************************************************/
#include <18f4550.h>
#device  adc=10
#device  pass_strings=in_ram
#device  *=16
/*****************************************************************************/
#FUSES NOWDT
#FUSES FCMEN                //Fail-safe clock monitor enabled
#FUSES IESO                 //Internal External Switch Over mode enabled
#FUSES PUT                  //Power Up Timer
#FUSES NOBROWNOUT               //No Brownout reset
#FUSES BORV43               //Brownout reset at 2.7V
#FUSES VREGEN               //USB voltage regulator enabled
#FUSES NOPBADEN             //PORTB pins are configured as digital I/O on RESET
#FUSES NOLPT1OSC            //Timer1 configured for higher power operation
#FUSES MCLR                 //Master Clear pin enabled
#FUSES STVREN               //Stack full/underflow will cause reset
#FUSES NOLVP                //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOXINST              //Extended set extension and Indexed Addressing mode disabled
(Legacy mode)
#FUSES NOPROTECT            //Code not protected from reading
#FUSES NOCPB                //No Boot Block code protection
#FUSES NOCPD                //No EE protection
#FUSES NOWRT                //Program memory not write protected
#FUSES NOWRTC               //Configuration registers not write protected
#FUSES NOWRTB               //Boot block not write protected
#FUSES NOWRTD               //Data EEPROM not write protected
#FUSES NOEBTR               //Memory not protected from table reads
#FUSES NOEBTRB              //Boot block not protected from table reads
/*****************************************************************************/
#use delay(clock=48MHz,crystal=4MHz,USB_FULL,restart_wdt)
/*****************************************************************************/
#use rs232(baud=115200,xmit=pin_d3,rcv=pin_d7,disable_ints,stream=ps)
#use rs232(uart1,baud=2400,receive_buffer=64,errors,stream=mdm)
#use rs232(baud=115200,xmit=pin_e0,rcv=pin_e2,enable=pin_e1,disable_ints,timeout=1
,errors,stream=kbd)
/*****************************************************************************/
#use fixed_io(a_outputs=pin_a1,pin_a2,pin_a3)
/*****************************************************************************
Librerias USB HID
```

```
*******************************************************************************/
#define USB_ISR_POLLING
#define USB_CABLE_IS_ATTACHED()  input(pin_d2)
#define USB_CONFIG_BUS_POWER 500
#define USB_CONFIG_HID_TX_SIZE 63
#define USB_CONFIG_HID_RX_SIZE 63
#include <pic18_usb.h>
#include <usb_desc_hid.h>
#include <usb.c>
/*******************************************************************************
Variables y constantes para el modem
*******************************************************************************/
#define   pwr_control pin_d5
#define   pwr_reset   pin_d4
/*******************************************************************************/
const char  id[]="smdu0001";
struct{
      enum{no_listo,listo}comando_at;
      enum{no_gsm,gsm}red;
      enum{no_gprs,gprs}acceso;
      enum{espera,error,ok}respuesta;
      enum{no_conectado,conectando,conectado}servidor;
      enum{no_disponible,disponible}cadena_modem;
      enum{antiguo,nuevo}comando;
      unsigned int8 n_sms;
      char index_sms[10];
      char buffer_comando[50];
     }modem;
/*******************************************************************************
Variables
*******************************************************************************/
unsigned int8  semaforo_usb=1;
unsigned int8  semaforo_modem=1;
unsigned int8  tiempo_reporte;
unsigned int8  enviar_sms;
char          nro_sms[14];
char clave1[10];
char clave2[10];
char clave3[10];
char clave [10];
struct{
       char estado;          //estado de la alarma
       char zona[8];
       char estado_zona[8];
       char pgm_zona[8];
       char disparo_evento_zona[8];
       char evento_zona[8];
```

```
    char retardo_zona[8];    //segundos
    char estado_pgm[4];
    char tiempo_pgm[4];      //segundos
    }alarma;
/*****************************************************************************
Constantes
*****************************************************************************/
#define led_gsm     pin_b5
#define led_gprs    pin_b6
#define led_servidor pin_b7
/*****************************************************************************/
#define out1 pin_c1
#define out2 pin_c2
#define out3 pin_d0
#define out4 pin_d1
/*****************************************************************************/
#define entrada_deshabilitada   0
#define entrada_habilitada      1
#define entrada_panico          2

#define desactivado             0
#define activado                1
#define activado_modo_presente  2

#define abierto                 0
#define cerrado                 1
#define corto                   2
/*****************************************************************************
Direcciones de las variables en la eeprom
*****************************************************************************/
#define ee_tiempo_reporte       getenv("EEPROM_ADDRESS")+0x00
#define ee_enviar_sms           getenv("EEPROM_ADDRESS")+0x01
#define ee_nro_sms              getenv("EEPROM_ADDRESS")+0x02
#define ee_clave1               getenv("EEPROM_ADDRESS")+0x10
#define ee_clave2               getenv("EEPROM_ADDRESS")+0x20
#define ee_clave3               getenv("EEPROM_ADDRESS")+0x30
#define ee_clave                getenv("EEPROM_ADDRESS")+0x90
#define ee_alarma_estado        getenv("EEPROM_ADDRESS")+0xa0
#define ee_alarma_zona0         getenv("EEPROM_ADDRESS")+0xa1
#define ee_alarma_zona1         getenv("EEPROM_ADDRESS")+0xa2
#define ee_alarma_zona2         getenv("EEPROM_ADDRESS")+0xa3
#define ee_alarma_zona3         getenv("EEPROM_ADDRESS")+0xa4
#define ee_alarma_zona4         getenv("EEPROM_ADDRESS")+0xa5
#define ee_alarma_zona5         getenv("EEPROM_ADDRESS")+0xa6
#define ee_alarma_zona6         getenv("EEPROM_ADDRESS")+0xa7
#define ee_alarma_zona7         getenv("EEPROM_ADDRESS")+0xa8
```

```
#define ee_alarma_pgm_zona0     getenv("EEPROM_ADDRESS")+0xa9
#define ee_alarma_pgm_zona1     getenv("EEPROM_ADDRESS")+0xaa
#define ee_alarma_pgm_zona2     getenv("EEPROM_ADDRESS")+0xab
#define ee_alarma_pgm_zona3     getenv("EEPROM_ADDRESS")+0xac
#define ee_alarma_pgm_zona4     getenv("EEPROM_ADDRESS")+0xad
#define ee_alarma_pgm_zona5     getenv("EEPROM_ADDRESS")+0xae
#define ee_alarma_pgm_zona6     getenv("EEPROM_ADDRESS")+0xaf
#define ee_alarma_pgm_zona7     getenv("EEPROM_ADDRESS")+0xb0
#define ee_alarma_disparo_evento_zona0  getenv("EEPROM_ADDRESS")+0xb1
#define ee_alarma_disparo_evento_zona1  getenv("EEPROM_ADDRESS")+0xb2
#define ee_alarma_disparo_evento_zona2  getenv("EEPROM_ADDRESS")+0xb3
#define ee_alarma_disparo_evento_zona3  getenv("EEPROM_ADDRESS")+0xb4
#define ee_alarma_disparo_evento_zona4  getenv("EEPROM_ADDRESS")+0xb5
#define ee_alarma_disparo_evento_zona5  getenv("EEPROM_ADDRESS")+0xb6
#define ee_alarma_disparo_evento_zona6  getenv("EEPROM_ADDRESS")+0xb7
#define ee_alarma_disparo_evento_zona7  getenv("EEPROM_ADDRESS")+0xb8
#define ee_alarma_retardo_zona0 getenv("EEPROM_ADDRESS")+0xb9
#define ee_alarma_retardo_zona1 getenv("EEPROM_ADDRESS")+0xba
#define ee_alarma_retardo_zona2 getenv("EEPROM_ADDRESS")+0xbb
#define ee_alarma_retardo_zona3 getenv("EEPROM_ADDRESS")+0xbc
#define ee_alarma_retardo_zona4 getenv("EEPROM_ADDRESS")+0xbd
#define ee_alarma_retardo_zona5 getenv("EEPROM_ADDRESS")+0xbe
#define ee_alarma_retardo_zona6 getenv("EEPROM_ADDRESS")+0xbf
#define ee_alarma_retardo_zona7 getenv("EEPROM_ADDRESS")+0xc0
#define ee_alarma_evento_zona0  getenv("EEPROM_ADDRESS")+0xc1
#define ee_alarma_evento_zona1  getenv("EEPROM_ADDRESS")+0xc2
#define ee_alarma_evento_zona2  getenv("EEPROM_ADDRESS")+0xc3
#define ee_alarma_evento_zona3  getenv("EEPROM_ADDRESS")+0xc4
#define ee_alarma_evento_zona4  getenv("EEPROM_ADDRESS")+0xc5
#define ee_alarma_evento_zona5  getenv("EEPROM_ADDRESS")+0xc6
#define ee_alarma_evento_zona6  getenv("EEPROM_ADDRESS")+0xc7
#define ee_alarma_evento_zona7  getenv("EEPROM_ADDRESS")+0xc8
#define ee_alarma_tiempo_pgm0   getenv("EEPROM_ADDRESS")+0xc9
#define ee_alarma_tiempo_pgm1   getenv("EEPROM_ADDRESS")+0xca
#define ee_alarma_tiempo_pgm2   getenv("EEPROM_ADDRESS")+0xcb
#define ee_alarma_tiempo_pgm3   getenv("EEPROM_ADDRESS")+0xcc
/*****************************************************************************
Valores por defecto de las variables en la eeprom
*****************************************************************************/
#rom ee_tiempo_reporte={1}
#rom ee_enviar_sms={0}
#rom ee_nro_sms={"70660715"}
#rom ee_clave1={"123"}
#rom ee_clave2={"456"}
#rom ee_clave3={"789"}
#rom ee_clave ={"1492"}
```

```
#rom  ee_alarma_estado={desactivado}
#rom  ee_alarma_zona0={entrada_habilitada}    //entrada habilitada
#rom  ee_alarma_zona1={entrada_deshabilitada}
#rom  ee_alarma_zona2={entrada_deshabilitada}
#rom  ee_alarma_zona3={entrada_deshabilitada}
#rom  ee_alarma_zona4={entrada_deshabilitada}
#rom  ee_alarma_zona5={entrada_deshabilitada}
#rom  ee_alarma_zona6={entrada_deshabilitada}
#rom  ee_alarma_zona7={entrada_deshabilitada}
#rom  ee_alarma_pgm_zona0={0}   //salida0 asociada a la zona 0
#rom  ee_alarma_pgm_zona1={0}
#rom  ee_alarma_pgm_zona2={0}
#rom  ee_alarma_pgm_zona3={0}
#rom  ee_alarma_pgm_zona4={0}   //salida0 asociada a la zona 4
#rom  ee_alarma_pgm_zona5={0}
#rom  ee_alarma_pgm_zona6={0}
#rom  ee_alarma_pgm_zona7={0}
#rom  ee_alarma_disparo_evento_zona0={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona1={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona2={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona3={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona4={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona5={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona6={activado<<4|abierto}
#rom  ee_alarma_disparo_evento_zona7={activado<<4|abierto}
#rom  ee_alarma_retardo_zona0={0}  //evento inmediato
#rom  ee_alarma_retardo_zona1={0}
#rom  ee_alarma_retardo_zona2={0}
#rom  ee_alarma_retardo_zona3={0}
#rom  ee_alarma_retardo_zona4={1}  //evento después de 1 segundos
#rom  ee_alarma_retardo_zona5={1}
#rom  ee_alarma_retardo_zona6={1}
#rom  ee_alarma_retardo_zona7={1}
#rom  ee_alarma_tiempo_pgm0={0}     //salida activa todo el tiempo
#rom  ee_alarma_tiempo_pgm1={1}
#rom  ee_alarma_tiempo_pgm2={1}
#rom  ee_alarma_tiempo_pgm3={1}     //salida activada 1s
/*****************************************************************************
Tareas del RTOS
*****************************************************************************/
#use rtos(timer=1,minor_cycle=1ms)
#task(rate=1ms,max=1ms,enabled=true)
void tarea_usb();
#task(rate=10ms,enabled=true)
void tarea_lectura_modem();
#task(rate=20ms,enabled=false)
void tarea_gestionar_salidas();
```

```
#task(rate=20ms,enabled=false)
void tarea_lectura_entradas();
#task(rate=100ms,enabled=false)
void tarea_mostrar_estados();
#task(rate=100ms,enabled=true)
void tarea_modem();
#task(rate=200ms,enabled=true)
void tarea_configuraciones();
#task(rate=1000ms,enabled=true)
void tarea_reporte();
/*************************************************************************
Librerías y funciones
*************************************************************************/
#include <string.h>
#include <stdlib.h>
#include <subrutinas_funciones.c>
/*************************************************************************/
```

El archivo subrutinas_funciones tiene la tarea de gestionar el manejo del modem GSM/GPRS

subrutinas_funciones.h

```
/*************************************************************************
Funciones y subrutinas para el modem GSM/GPRS
*************************************************************************/
void modem_off()
{
  output_low(pwr_reset);
  delay_ms(200);
  output_float(pwr_reset);
}
void modem_on()
{
  modem.red=no_gsm;
  modem.acceso=no_gprs;
  modem.servidor=no_conectado;
  //modem.comando=antiguo;
  modem.comando_at=no_listo;
  modem.respuesta=error;
  output_low(pwr_control);
  modem.n_sms=0;
  delay_ms(200);
  output_float(pwr_control);
}
/*************************************************************************/
void actualizar_estados_modem(char *buffer_rx_modem)
{
  unsigned int8 *p;
  if(strstr(buffer_rx_modem,"OK"))
```

```
  {
    modem.respuesta=ok;
    return;
  }
  if(strstr(buffer_rx_modem,"ERROR"))
  {
    modem.respuesta=error;
    modem.servidor=no_conectado;
    return;
  }
  if(strstr(buffer_rx_modem,"AT-Command Interpreter ready"))
  {
    modem.comando_at=listo;
    return;
  }
  if(strstr(buffer_rx_modem,"CONNECT"))
  {
    output_high(led_servidor);
    modem.servidor=conectado;
    return;
  }
  if(strstr(buffer_rx_modem,"NO CARRIER"))
  {
    output_low(led_servidor);
    modem.servidor=no_conectado;
    return;
  }
  p=strstr(buffer_rx_modem,"+CREG: ");
  if(p)
  {
    p+=7;
    if((*p=='1')||(*p=='5'))
    {
      output_high(led_gsm);
      modem.red=gsm;
      fprintf(ps,"\r\nModem registrado en la red GSM\r\n");
      if(usb_enumerated())
      {
        usb_put_packet(USB_HID_ENDPOINT,"Modem registrado en la red
GSM",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
      }
    }
    else
    {
      output_low(led_gsm);
      modem.red=no_gsm;
      fprintf(ps,"\r\nModem sin registro en la red GSM\r\n");
```

```
      if(usb_enumerated())
      {
        usb_put_packet(USB_HID_ENDPOINT,"Modem sin registro en la red
GSM",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
      }
    }
    return;
  }
  p=strstr(buffer_rx_modem,"+CGREG: ");
  if(p)
  {
    p+=8;
    if((*p=='1')||(*p=='5'))
    {
      output_high(led_gprs);
      modem.acceso=gprs;
      fprintf(ps,"\r\nModem registrado en la red GPRS\r\n");
      if(usb_enumerated())
      {
        usb_put_packet(USB_HID_ENDPOINT,"Modem registrado en la red
GPRS",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
      }
    }
    else
    {
      output_low(led_gprs);
      modem.acceso=no_gprs;
      fprintf(ps,"\r\nModem sin registro en la red GPRS\r\n");
      if(usb_enumerated())
      {
        usb_put_packet(USB_HID_ENDPOINT,"Modem sin registro en la red
GPRS",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
      }
    }
  }
  p=strstr(buffer_rx_modem,"+CMGL: ");
  if(p)
  {
    p+=7;
    *strstr(buffer_rx_modem,',')=0;
    modem.index_sms[modem.n_sms++]=atoi(p);
    return;
  }
  p=strstr(buffer_rx_modem,"+CMTI: \"SM\",");
  if(p)
  {
    p+=12;
```

```
    *strstr(buffer_rx_modem,',')=0;
    modem.index_sms[modem.n_sms++]=atoi(p);
    return;
  }
  p=strstr(buffer_rx_modem,id);
  if(p)
  {
    strcpy(modem.buffer_comando,p);
    modem.comando=nuevo;
  }
}
/*****************************************************************************/
```

Código fuente, se encarga de gestionar todas las tareas del equipo de monitoreo.

control.c

```
#include <control.h>
/*****************************************************************************/
void tarea_usb(){usb_task();}
/*****************************************************************************/
void tarea_lectura_modem()
{
  char  rx;
  static char  buffer_rx[USB_CONFIG_HID_TX_SIZE+1];
  static unsigned int8 n;
  while(kbhit(mdm))
  {
    rx=fgetc(mdm);
    fputc(rx,ps);
    if(isprint(rx))
    {
      buffer_rx[n++]=rx;
    }
    if((n==USB_CONFIG_HID_TX_SIZE)||(rx=='\n'))
    {
      buffer_rx[n]='\0';
      n=0;
      if(usb_enumerated())
      {
        rtos_wait(semaforo_usb);

usb_put_packet(USB_HID_ENDPOINT,buffer_rx,USB_CONFIG_HID_TX_SIZE,USB_DTS_TO
GGLE);
        rtos_yield();
        rtos_signal(semaforo_usb);
      }
      actualizar_estados_modem(buffer_rx);
    }
  }
```

```
}
/*****************************************************************************/
void tarea_modem()
{
  unsigned int8 n;
  unsigned int8 *p;
  for(n=0;n<20;rtos_yield(),n++);
  rtos_enable(tarea_lectura_entradas);
  rtos_enable(tarea_gestionar_salidas);
  rtos_enable(tarea_mostrar_estados);
  fprintf(ps,"\r\nIniciando\r\n");
  if(usb_enumerated())
  {
    rtos_wait(semaforo_usb);

usb_put_packet(USB_HID_ENDPOINT,"Iniciando",USB_CONFIG_HID_TX_SIZE,USB_DTS_
TOGGLE);
    rtos_yield();
    rtos_signal(semaforo_usb);
  }
  modem_on();
  rtos_await(modem.comando_at==listo);
  fprintf(ps,"Modem activo\r\n");
  if(usb_enumerated())
  {
    rtos_wait(semaforo_usb);
    usb_put_packet(USB_HID_ENDPOINT,"Modem
activo",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    rtos_signal(semaforo_usb);
  }
  modem.respuesta=espera;
  rtos_wait(semaforo_modem);
  fprintf(mdm,"ate1\r");
  rtos_await(modem.respuesta==ok);
  rtos_signal(semaforo_modem);
  fprintf(ps,"Modem_en linea\r\n");
  if(usb_enumerated())
  {
    rtos_wait(semaforo_usb);
    usb_put_packet(USB_HID_ENDPOINT,"Modem_en
linea",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    rtos_signal(semaforo_usb);
  }
  while(true)
  {
```

```
if(modem.servidor==no_conectado)
{
  do
  {
    for(n=0;n<50;rtos_yield(),n++);
    rtos_wait(semaforo_modem);
    modem.respuesta=espera;
    modem.n_sms=0;
    fprintf(mdm,"at+cmgl=\"ALL\"\r");
    rtos_await(modem.respuesta!=espera);
    rtos_signal(semaforo_modem);
  }
  while(modem.respuesta==error);
  modem.comando=antiguo;
  if(modem.n_sms>0)
  {
    for(n=0;n<modem.n_sms;n++)
    {
      do
      {
        rtos_wait(semaforo_modem);
        modem.respuesta=espera;
        fprintf(mdm,"at+cmgr=%u\r",modem.index_sms[n]);
        rtos_await(modem.respuesta!=espera);
        rtos_signal(semaforo_modem);
      }
      while(modem.respuesta==error);
      if(modem.comando==nuevo)
      {
        modem.comando=antiguo;
        fprintf(ps,"comando:%s\r\n",modem.buffer_comando);
        p=modem.buffer_comando+8;
        switch(*p)
        {
          case
'0':{output_bit(out1,(*++p)&0x01);alarma.estado_pgm[0]=(*p)&0x0f;break;}
          case
'1':{output_bit(out2,(*++p)&0x01);alarma.estado_pgm[1]=(*p)&0x0f;break;}
          case
'2':{output_bit(out3,(*++p)&0x01);alarma.estado_pgm[2]=(*p)&0x0f;break;}
          case
'3':{output_bit(out4,(*++p)&0x01);alarma.estado_pgm[3]=(*p)&0x0f;break;}
          case '4':
          {
            alarma.estado=(*++p)&0x03;
            if(alarma.estado==desactivado)
            {
```

```
                alarma.estado_pgm[0]=0;
                alarma.estado_pgm[1]=0;
                alarma.estado_pgm[2]=0;
                alarma.estado_pgm[3]=0;
                output_low(out1);
                output_low(out2);
                output_low(out3);
                output_low(out4);
                alarma.evento_zona[0]=0;
                alarma.evento_zona[1]=0;
                alarma.evento_zona[2]=0;
                alarma.evento_zona[3]=0;
                alarma.evento_zona[4]=0;
                alarma.evento_zona[5]=0;
                alarma.evento_zona[6]=0;
                alarma.evento_zona[7]=0;
              }
            }
          }
        }
        do
        {
          rtos_wait(semaforo_modem);
          modem.respuesta=espera;
          fprintf(mdm,"at+cmgd=%u\r",modem.index_sms[n]);
          rtos_await(modem.respuesta!=espera);
          rtos_signal(semaforo_modem);
        }
        while(modem.respuesta==error);
      }
    }
    if((modem.red==gsm)&&(modem.acceso==gprs))
    {
      fprintf(ps,"\r\nConectando al servidor\r\n");
      if(usb_enumerated())
      {
        rtos_wait(semaforo_usb);
        usb_put_packet(USB_HID_ENDPOINT,"Conectando al
servidor",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
        rtos_signal(semaforo_usb);
      }
      rtos_wait(semaforo_modem);
      modem.servidor=conectando;
      fprintf(mdm,"atd*99***1#\r");
      rtos_signal(semaforo_modem);
    }
  }
```

```
    if(modem.comando==nuevo)
    {
      modem.comando=antiguo;
      fprintf(ps,"comando:%s\r\n",modem.buffer_comando);
      p=modem.buffer_comando+8;
      switch(*p)
      {
        case '0':{output_bit(out1,(*++p)&0x01);alarma.estado_pgm[0]=(*p)&0x0f;break;}
        case '1':{output_bit(out2,(*++p)&0x01);alarma.estado_pgm[1]=(*p)&0x0f;break;}
        case '2':{output_bit(out3,(*++p)&0x01);alarma.estado_pgm[2]=(*p)&0x0f;break;}
        case '3':{output_bit(out4,(*++p)&0x01);alarma.estado_pgm[3]=(*p)&0x0f;break;}
        case '4':
        {
          alarma.estado=(*++p)&0x03;
          if(alarma.estado==desactivado)
          {
            alarma.estado_pgm[0]=0;
            alarma.estado_pgm[1]=0;
            alarma.estado_pgm[2]=0;
            alarma.estado_pgm[3]=0;
            output_low(out1);
            output_low(out2);
            output_low(out3);
            output_low(out4);
            alarma.evento_zona[0]=0;
            alarma.evento_zona[1]=0;
            alarma.evento_zona[2]=0;
            alarma.evento_zona[3]=0;
            alarma.evento_zona[4]=0;
            alarma.evento_zona[5]=0;
            alarma.evento_zona[6]=0;
            alarma.evento_zona[7]=0;
          }
        }
      }
    }
    rtos_yield();
  }
}
/*****************************************************************************/
void tarea_reporte()
{
  unsigned int8 contador;
  unsigned int8 aux=0;
  char buffer_reporte[32];
  char buffer_aux[3];
  static unsigned int8 estado_alarma;
```

```
  static unsigned int8 evento_zona[8];
  static unsigned int8 estado_pgm[4];
  for(contador=0;contador<60*tiempo_reporte;contador++)
  {
    if(estado_alarma!=alarma.estado){estado_alarma=alarma.estado;aux=1;}
if(evento_zona[0]!=alarma.evento_zona[0]){evento_zona[0]=alarma.evento_zona[0];aux=1;}
if(evento_zona[1]!=alarma.evento_zona[1]){evento_zona[1]=alarma.evento_zona[1];aux=1;}
if(evento_zona[2]!=alarma.evento_zona[2]){evento_zona[2]=alarma.evento_zona[2];aux=1;}
if(evento_zona[3]!=alarma.evento_zona[3]){evento_zona[3]=alarma.evento_zona[3];aux=1;}
if(evento_zona[4]!=alarma.evento_zona[4]){evento_zona[4]=alarma.evento_zona[4];aux=1;}
if(evento_zona[5]!=alarma.evento_zona[5]){evento_zona[5]=alarma.evento_zona[5];aux=1;}
if(evento_zona[6]!=alarma.evento_zona[6]){evento_zona[6]=alarma.evento_zona[6];aux=1;}
if(evento_zona[7]!=alarma.evento_zona[7]){evento_zona[7]=alarma.evento_zona[7];aux=1;}
if(estado_pgm[0]!=alarma.estado_pgm[0]){estado_pgm[0]=alarma.estado_pgm[0];aux=1;}
if(estado_pgm[1]!=alarma.estado_pgm[1]){estado_pgm[1]=alarma.estado_pgm[1];aux=1;}
if(estado_pgm[2]!=alarma.estado_pgm[2]){estado_pgm[2]=alarma.estado_pgm[2];aux=1;}
if(estado_pgm[3]!=alarma.estado_pgm[3]){estado_pgm[3]=alarma.estado_pgm[3];aux=1;}
    if(aux==1){break;}
    rtos_yield();
  }
  sprintf(buffer_reporte,"%s%u",id,alarma.estado);
  for(contador=0;contador<8;contador++)
  {
if((alarma.zona[contador]==entrada_habilitada)||(alarma.zona[contador]==entrada_panico))
    {
      sprintf(buffer_aux,"%u",alarma.estado_zona[contador]);
      strcat(buffer_reporte,buffer_aux);
    }
    else
    {
      strcat(buffer_reporte,"3");//entrada desabilitada
    }
  }
  for(contador=0;contador<8;contador++)
  {
    sprintf(buffer_aux,"%u",alarma.evento_zona[contador]);
    strcat(buffer_reporte,buffer_aux);
  }
  for(contador=0;contador<4;contador++)
  {
    sprintf(buffer_aux,"%u",alarma.estado_pgm[contador]);
    strcat(buffer_reporte,buffer_aux);
  }
  if(modem.servidor==conectando)
  {
    rtos_wait(semaforo_modem);
    fprintf(mdm,"+++");
```

```
    rtos_await(modem.servidor!=conectando);
    rtos_signal(semaforo_modem);
  }
  else if(modem.servidor==conectado)
  {
    fprintf(ps,"%sG\r\n",buffer_reporte);
    fprintf(mdm,"%s\r",buffer_reporte);
    if(usb_enumerated())
    {
      rtos_wait(semaforo_usb);
usb_put_packet(USB_HID_ENDPOINT,strcat(buffer_reporte,"G\r\n"),USB_CONFIG_HID_TX_
SIZE,USB_DTS_TOGGLE);
      rtos_signal(semaforo_usb);
    }
  }
  else if((modem.red==gsm)&&(enviar_sms==1))
  {
    fprintf(ps,"%sS\r\n",buffer_reporte);
    rtos_wait(semaforo_modem);
    fprintf(ps,"ini sms\r\n");
    fprintf(mdm,"at+cmgs=\"%s\"\r",nro_sms);
    fprintf(ps,"sms\r\n");
    while(getc(mdm)!='>');
    fprintf(mdm,"%s%c",buffer_reporte,26);
    rtos_signal(semaforo_modem);
    if(usb_enumerated())
    {
      rtos_wait(semaforo_usb);
usb_put_packet(USB_HID_ENDPOINT,strcat(buffer_reporte,"S\r\n"),USB_CONFIG_HID_TX_
SIZE,USB_DTS_TOGGLE);
      rtos_signal(semaforo_usb);
    }
  }
}
/*****************************************************************************/
void tarea_configuraciones()
{
  char buffer[USB_CONFIG_HID_TX_SIZE+1];
  char buffer_usb_rx[50];
  char contador;
  char *p,*s;
  rtos_await(usb_kbhit(USB_HID_ENDPOINT));
  usb_get_packet(USB_HID_ENDPOINT,buffer_usb_rx, USB_CONFIG_HID_RX_SIZE);
  fprintf(ps,"\r\n%s\r\n",buffer_usb_rx);
  rtos_wait(semaforo_usb);
usb_put_packet(USB_HID_ENDPOINT,buffer_usb_rx,USB_CONFIG_HID_TX_SIZE,USB_DTS
_TOGGLE);
```

```
  rtos_yield();
  rtos_signal(semaforo_usb);
  if(modem.servidor!=no_conectado)
  {
    fprintf(ps,"\r\nEspere un momento\r\n");
    rtos_wait(semaforo_usb);
    usb_put_packet(USB_HID_ENDPOINT,"Espere un
momento",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    rtos_signal(semaforo_usb);
    rtos_wait(semaforo_modem);
    fprintf(mdm,"+++");
    rtos_await(modem.servidor!=conectando);
    rtos_signal(semaforo_modem);
  }
  if(strstr(buffer_usb_rx,"!cmd monitoreo"))
  {
    fputs("************************",ps);
    fputs("*   Modo monitoreo     *",ps);
    fputs("************************",ps);
    rtos_wait(semaforo_usb);
usb_put_packet(USB_HID_ENDPOINT,"************************",USB_CONFIG_HID_T
X_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    usb_put_packet(USB_HID_ENDPOINT,"*   Modo monitoreo
*",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
usb_put_packet(USB_HID_ENDPOINT,"************************",USB_CONFIG_HID_T
X_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    rtos_signal(semaforo_usb);
    rtos_enable(tarea_modem);
    rtos_enable(tarea_reporte);
    rtos_enable(tarea_lectura_entradas);
  }
  else if(!strcmp(buffer_usb_rx,"!cmd consola"))
  {
    fputs("************************",ps);
    fputs("*     Modo consola     *",ps);
    fputs("************************",ps);
    rtos_wait(semaforo_usb);
usb_put_packet(USB_HID_ENDPOINT,"************************",USB_CONFIG_HID_T
X_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
    usb_put_packet(USB_HID_ENDPOINT,"*     Modo consola
*",USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
    rtos_yield();
```

```
usb_put_packet(USB_HID_ENDPOINT,"************************",USB_CONFIG_HID_T
X_SIZE,USB_DTS_TOGGLE);
      rtos_yield();
      rtos_signal(semaforo_usb);
      rtos_disable(tarea_modem);
      rtos_disable(tarea_reporte);
      rtos_disable(tarea_lectura_entradas);
    }
    else if(!strncmp(buffer_usb_rx,"apn=",4))
    {
      p=buffer_usb_rx+4;
      fprintf(mdm,"at+cgdcont=1,\"ip\",\"%s\",\"\",0,0\r",p);
    }
    else if(!strncmp(buffer_usb_rx,"ip servidor=",12))
    {
      p=buffer_usb_rx+12;
      s=strchr(buffer_usb_rx,',');
      *s=0;
      fprintf(mdm,"at$paddst=\"%s\",%s\r",p,++s);
      rtos_yield();
      fprintf(mdm,"at$friend=1,1,\"%s\"\r",p);
    }
    else if(!strncmp(buffer_usb_rx,"reporte=",8))
    {
      p=buffer_usb_rx+8;
      tiempo_reporte=make8(atol(p),0);
    }
    else if(!strncmp(buffer_usb_rx,"sms=",4))
    {
      p=buffer_usb_rx+4;
      enviar_sms=make8(atol(p),0);
    }
    else if(!strncmp(buffer_usb_rx,"nro sms=",8))
    {
      p=buffer_usb_rx+8;
      strcpy(nro_sms,p);
    }
    else if(!strncmp(buffer_usb_rx,"clave1=",7))
    {
      p=buffer_usb_rx+7;
      strcpy(clave1,p);
    }
    else if(!strncmp(buffer_usb_rx,"clave2=",7))
    {
      p=buffer_usb_rx+7;
      strcpy(clave2,p);
    }
```

```
else if(!strncmp(buffer_usb_rx,"clave3=",7))
{
  p=buffer_usb_rx+7;
  strcpy(clave3,p);
}
else if(!strncmp(buffer_usb_rx,"zona",4))
{
  p=buffer_usb_rx+4;
  if(*p>'8')
  {
    return;
  }
  alarma.zona[((*p)&0x0f)]=make8(atol(p+2),0);
}
else if(!strncmp(buffer_usb_rx,"pgm zona",8))
{
  p=buffer_usb_rx+8;
  if(*p>'8')
  {
    return;
  }
  alarma.pgm_zona[((*p)&0x0f)]=make8(atol(p+2),0);
}
else if(!strncmp(buffer_usb_rx,"disparo evento zona",19))
{
  p=buffer_usb_rx+19;
  if(*p>'8')
  {
    return;
  }
  alarma.disparo_evento_zona[((*p)&0x0f)]=((*(p+2)&0x0f)<<4)|(*(p+3)&0x0f);
}
else if(!strncmp(buffer_usb_rx,"retardo de zona",15))
{
  p=buffer_usb_rx+15;
  if(*p>'8')
  {
    return;
  }
  alarma.retardo_zona[((*p)&0x0f)]=make8(atol(p+2),0);
}
else if(!strncmp(buffer_usb_rx,"tiempo activo pgm",17))
{
  p=buffer_usb_rx+17;
  if(*p>'4')
  {
    return;
```

```
    }
    alarma.tiempo_pgm[((*p)&0x0f)]=make8(atol(p+2),0);
  }
  else if(!strncmp(buffer_usb_rx,"parametros",10))
  {
    sprintf(buffer,"reporte=%u",tiempo_reporte);

usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);
    sprintf(buffer,"sms=%u",enviar_sms);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);
    sprintf(buffer,"nro sms=%s",nro_sms);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);
    sprintf(buffer,"clave1=%s",clave1);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);
    sprintf(buffer,"clave2=%s",clave2);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);
    sprintf(buffer,"clave3=%s",clave3);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
    rtos_yield();
    fprintf(ps,"%s\r\n",buffer);

    for(contador=0;contador<8;contador++)
    {
      sprintf(buffer,"zona%u=%u",contador,alarma.zona[contador]);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
      rtos_yield();
      fprintf(ps,"%s\r\n",buffer);
    }
    for(contador=0;contador<8;contador++)
    {
```

```
        sprintf(buffer,"pgm zona%u=%u",contador,alarma.pgm_zona[contador]);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
        rtos_yield();
        fprintf(ps,"%s\r\n",buffer);
      }
      for(contador=0;contador<8;contador++)
      {
        sprintf(buffer,"disparo evento
zona%u=%x",contador,alarma.disparo_evento_zona[contador]);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
        rtos_yield();
        fprintf(ps,"%s\r\n",buffer);
      }
      for(contador=0;contador<8;contador++)
      {
        sprintf(buffer,"retardo de zona%u=%u",contador,alarma.retardo_zona[contador]);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
        rtos_yield();
        fprintf(ps,"%s\r\n",buffer);
      }
      for(contador=0;contador<4;contador++)
      {
        sprintf(buffer,"tiempo activo pgm%u=%u",contador,alarma.tiempo_pgm[contador]);
usb_put_packet(USB_HID_ENDPOINT,buffer,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGG
LE);
        rtos_yield();
        fprintf(ps,"%s\r\n",buffer);
      }
      fprintf(mdm,"at&v\r");
    }
    else if(!strncmp(buffer_usb_rx,"guardar",7))
    {
      write_eeprom(ee_tiempo_reporte-getenv("EEPROM_ADDRESS"),tiempo_reporte);
      write_eeprom(ee_enviar_sms-getenv("EEPROM_ADDRESS"),enviar_sms);

      contador=0;
do{write_eeprom(ee_nro_sms+contador,nro_sms[contador]);}while(nro_sms[contador++]!=0)
;
      contador=0;
do{write_eeprom(ee_clave1+contador,clave1[contador]);}while(clave1[contador++]!=0);
      contador=0;
do{write_eeprom(ee_clave2+contador,clave2[contador]);}while(clave2[contador++]!=0);
      contador=0;
do{write_eeprom(ee_clave3+contador,clave3[contador]);}while(clave3[contador++]!=0);
```

```
    write_eeprom(ee_alarma_estado-getenv("EEPROM_ADDRESS"),alarma.estado);
    write_eeprom(ee_alarma_zona0-getenv("EEPROM_ADDRESS"),alarma.zona[0]);
    write_eeprom(ee_alarma_zona1-getenv("EEPROM_ADDRESS"),alarma.zona[1]);
    write_eeprom(ee_alarma_zona2-getenv("EEPROM_ADDRESS"),alarma.zona[2]);
    write_eeprom(ee_alarma_zona3-getenv("EEPROM_ADDRESS"),alarma.zona[3]);
    write_eeprom(ee_alarma_zona4-getenv("EEPROM_ADDRESS"),alarma.zona[4]);
    write_eeprom(ee_alarma_zona5-getenv("EEPROM_ADDRESS"),alarma.zona[5]);
    write_eeprom(ee_alarma_zona6-getenv("EEPROM_ADDRESS"),alarma.zona[6]);
    write_eeprom(ee_alarma_zona7-getenv("EEPROM_ADDRESS"),alarma.zona[7]);
    write_eeprom(ee_alarma_pgm_zona0-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[0]);
    write_eeprom(ee_alarma_pgm_zona1-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[1]);
    write_eeprom(ee_alarma_pgm_zona2-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[2]);
    write_eeprom(ee_alarma_pgm_zona3-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[3]);
    write_eeprom(ee_alarma_pgm_zona4-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[4]);
    write_eeprom(ee_alarma_pgm_zona5-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[5]);
    write_eeprom(ee_alarma_pgm_zona6-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[6]);
    write_eeprom(ee_alarma_pgm_zona7-
getenv("EEPROM_ADDRESS"),alarma.pgm_zona[7]);
    write_eeprom(ee_alarma_disparo_evento_zona0-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[0]);
    write_eeprom(ee_alarma_disparo_evento_zona1-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[1]);
    write_eeprom(ee_alarma_disparo_evento_zona2-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[2]);
    write_eeprom(ee_alarma_disparo_evento_zona3-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[3]);
    write_eeprom(ee_alarma_disparo_evento_zona4-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[4]);
    write_eeprom(ee_alarma_disparo_evento_zona5-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[5]);
    write_eeprom(ee_alarma_disparo_evento_zona6-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[6]);
    write_eeprom(ee_alarma_disparo_evento_zona7-
getenv("EEPROM_ADDRESS"),alarma.disparo_evento_zona[7]);
    write_eeprom(ee_alarma_retardo_zona0-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[0]);
    write_eeprom(ee_alarma_retardo_zona1-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[1]);
    write_eeprom(ee_alarma_retardo_zona2-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[2]);
```

```
      write_eeprom(ee_alarma_retardo_zona3-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[3]);
      write_eeprom(ee_alarma_retardo_zona4-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[4]);
      write_eeprom(ee_alarma_retardo_zona5-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[5]);
      write_eeprom(ee_alarma_retardo_zona6-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[6]);
      write_eeprom(ee_alarma_retardo_zona7-
getenv("EEPROM_ADDRESS"),alarma.retardo_zona[7]);
      write_eeprom(ee_alarma_tiempo_pgm0-
getenv("EEPROM_ADDRESS"),alarma.tiempo_pgm[0]);
      write_eeprom(ee_alarma_tiempo_pgm1-
getenv("EEPROM_ADDRESS"),alarma.tiempo_pgm[1]);
      write_eeprom(ee_alarma_tiempo_pgm2-
getenv("EEPROM_ADDRESS"),alarma.tiempo_pgm[2]);
      write_eeprom(ee_alarma_tiempo_pgm3-
getenv("EEPROM_ADDRESS"),alarma.tiempo_pgm[3]);
      fprintf(mdm,"at+cgreg=1\r");
      rtos_yield();
      fprintf(mdm,"at+creg=1\r");
      rtos_yield();
      fprintf(mdm,"at$areg=1\r");
      rtos_yield();
      fprintf(mdm,"at%%cgaatt=0,1\r");
      rtos_yield();
      fprintf(mdm,"at+cmee=2\r");
      rtos_yield();
      fprintf(mdm,"at+crc=1\r");
      rtos_yield();
      fprintf(mdm,"at&w\r");
      rtos_yield();
      disable_interrupts(int_rda);
      disable_interrupts(GLOBAL);
      usb_detach();
      delay_ms(4000);
      reset_cpu();
   }
   else
   {
      fprintf(mdm,"%s\r",buffer_usb_rx);
   }
}
/*****************************************************************************/
void tarea_lectura_entradas()
{
   static unsigned int8 bandera_disparo_zona[8];
```

```
static unsigned int8 contador_retardo_zona[8];
static unsigned int8 retardo_zona[8];
char  rx_kbd[10];
unsigned int16 dato_adc;
unsigned int8 canal_adc;
unsigned int8 n=0;
static char bandera_codigo;
fprintf(kbd,"1\r");
fgets(rx_kbd,kbd);
*strchr(rx_kbd,'\r')=0;
if((strcmp(clave1,rx_kbd)==0)||(strcmp(clave2,rx_kbd)==0)||(strcmp(clave3,rx_kbd)==0))
{
  rtos_disable(tarea_mostrar_estados);
  bandera_codigo=1;
  fprintf(kbd,"2\r");
}
else if((strcmp("1",rx_kbd)==0)&&(bandera_codigo==1))
{
  bandera_codigo=0;
  alarma.estado=activado;
  rtos_enable(tarea_mostrar_estados);
}
else if((strcmp("2",rx_kbd)==0)&&(bandera_codigo==1))
{
  alarma.estado=desactivado;
  alarma.estado_pgm[0]=0;
  alarma.estado_pgm[1]=0;
  alarma.estado_pgm[2]=0;
  alarma.estado_pgm[3]=0;
  output_low(out1);
  output_low(out2);
  output_low(out3);
  output_low(out4);
  alarma.evento_zona[0]=0;
  alarma.evento_zona[1]=0;
  alarma.evento_zona[2]=0;
  alarma.evento_zona[3]=0;
  alarma.evento_zona[4]=0;
  alarma.evento_zona[5]=0;
  alarma.evento_zona[6]=0;
  alarma.evento_zona[7]=0;
  bandera_codigo=0;
  rtos_enable(tarea_mostrar_estados);
}
else if((strcmp("3",rx_kbd)==0)&&(bandera_codigo==1))
{
  alarma.estado=activado_modo_presente;
```

```
    bandera_codigo=0;
    rtos_enable(tarea_mostrar_estados);
  }
  else if(strcmp("nan",rx_kbd)!=0)
  {
    rtos_disable(tarea_mostrar_estados);
    bandera_codigo=0;
    fprintf(kbd,"4\r");
  }
  for(canal_adc=0;canal_adc<8;canal_adc++)
  {
if((alarma.zona[canal_adc]==entrada_habilitada)||(alarma.zona[canal_adc]==entrada_panico
))
    {
      output_a(canal_adc*2);
      rtos_yield();
      dato_adc=read_adc(ADC_START_AND_READ);
      if(600<dato_adc)
      {
        alarma.estado_zona[canal_adc]=abierto;
      }
      else if((600>=dato_adc)&&(dato_adc>=200))
      {
        alarma.estado_zona[canal_adc]=cerrado;
      }
      else
      {
        alarma.estado_zona[canal_adc]=corto;
      }
if((alarma.zona[canal_adc]==entrada_panico)&&((alarma.disparo_evento_zona[canal_adc]&
0x0f)==alarma.estado_zona[canal_adc])              )
      {
        bandera_disparo_zona[canal_adc]=1;
      }
if(alarma.disparo_evento_zona[canal_adc]==((alarma.estado<<4)|alarma.estado_zona[canal_
adc]))
      {
        bandera_disparo_zona[canal_adc]=1;
      }
    }
  }
  for(n=0;n<8;n++)
  {
    if(bandera_disparo_zona[n])
    {
      if(retardo_zona[n]==alarma.retardo_zona[n])
      {
```

```
        retardo_zona[n]=0;
        bandera_disparo_zona[n]=0;
        alarma.evento_zona[n]=1;
      }
      else if(++contador_retardo_zona[n]==50)   // 1 segundo
      {
        contador_retardo_zona[n]=0;
        retardo_zona[n]++;
      }
    }
  }
}
void tarea_gestionar_salidas()
{
  static unsigned int8 contador_tiempo_pgm[4];
  static unsigned int8 tiempo_pgm[4];
  static unsigned int8 n;
  for(n=0;n<8;n++)
  {
    switch(alarma.evento_zona[n])
    {
      case 1:
      {
        switch(alarma.pgm_zona[n])
        {
          case 0:
          {
            output_high(out1);
            alarma.estado_pgm[0]=activado;
            alarma.evento_zona[n]++;
            break;
          }
          case 1:
          {
            output_high(out2);
            alarma.estado_pgm[1]=activado;
            alarma.evento_zona[n]++;
            break;
          }
          case 2:
          {
            output_high(out3);
            alarma.estado_pgm[2]=activado;
            alarma.evento_zona[n]++;
            break;
          }
          case 3:
```

```
            {
              output_high(out4);
              alarma.estado_pgm[3]=activado;
              alarma.evento_zona[n]++;
            }
          }
          break;
        }
        case 2:
        {
          if(++contador_tiempo_pgm[alarma.pgm_zona[n]]==100)
          {
            contador_tiempo_pgm[alarma.pgm_zona[n]]=0;
if(++tiempo_pgm[alarma.pgm_zona[n]]==alarma.tiempo_pgm[alarma.pgm_zona[n]])
            {
              alarma.evento_zona[n]=0;
              tiempo_pgm[alarma.pgm_zona[n]]=0;
              switch(alarma.pgm_zona[n])
              {
                case 0:
                {
                  output_low(out1);
                  alarma.estado_pgm[0]=desactivado;
                  break;
                }
                case 1:
                {
                  output_low(out2);
                  alarma.estado_pgm[1]=desactivado;
                  break;
                }
                case 2:
                {
                  output_low(out3);
                  alarma.estado_pgm[2]=desactivado;
                  break;
                }
                case 3:
                {
                  output_low(out4);
                  alarma.estado_pgm[3]=desactivado;
                }
              }
            }
          }
        }
      }
    }
```

```
  }
}
void tarea_mostrar_estados()
{
  unsigned int n;
  fprintf(kbd,"3");
  fprintf(kbd,"%c",alarma.estado);
  for(n=0;n<8;n++)
  {
    if((alarma.zona[n]==entrada_habilitada)||(alarma.zona[n]==entrada_panico))
    {
      fprintf(kbd,"%c",alarma.estado_zona[n]);
    }
    else
    {
      fprintf(kbd,"%c",3);
    }
  }
  for(n=0;n<4;n++)
  {
    fprintf(kbd,"%c",alarma.estado_pgm[n]);
  }
  fprintf(kbd,"\r");
}
/*****************************************************************************
Rutina principal
*****************************************************************************/
void main()
{
  unsigned int8 n=0;
  modem_off();
  output_low(led_gsm);
  output_low(led_gprs);
  output_low(led_servidor);
  while(!setup_oscillator(OSC_STATE_EXT_RUNNING));
  while(!setup_oscillator(OSC_STATE_STABLE));

  output_low(out1);
  output_low(out2);
  output_low(out3);
  output_low(out4);
  alarma.estado_pgm[0]=desactivado;
  alarma.estado_pgm[1]=desactivado;
  alarma.estado_pgm[2]=desactivado;
  alarma.estado_pgm[3]=desactivado;
  output_low(out1);
  output_low(out2);
```

```
output_low(out3);
output_low(out4);
alarma.evento_zona[0]=0;
alarma.evento_zona[1]=0;
alarma.evento_zona[2]=0;
alarma.evento_zona[3]=0;
alarma.evento_zona[4]=0;
alarma.evento_zona[5]=0;
alarma.evento_zona[6]=0;
alarma.evento_zona[7]=0;
alarma.estado_zona[0]=0;
alarma.estado_zona[1]=0;
alarma.estado_zona[2]=0;
alarma.estado_zona[3]=0;
alarma.estado_zona[4]=0;
alarma.estado_zona[5]=0;
alarma.estado_zona[6]=0;
alarma.estado_zona[7]=0
tiempo_reporte=read_eeprom(ee_tiempo_reporte);
enviar_sms=read_eeprom(ee_enviar_sms);
do{nro_sms[n]=read_eeprom(ee_nro_sms+n);}while(nro_sms[n++]!='\0');
alarma.estado=read_eeprom(ee_alarma_estado);
alarma.zona[0]=read_eeprom(ee_alarma_zona0);
alarma.zona[1]=read_eeprom(ee_alarma_zona1);
alarma.zona[2]=read_eeprom(ee_alarma_zona2);
alarma.zona[3]=read_eeprom(ee_alarma_zona3);
alarma.zona[4]=read_eeprom(ee_alarma_zona4);
alarma.zona[5]=read_eeprom(ee_alarma_zona5);
alarma.zona[6]=read_eeprom(ee_alarma_zona6);
alarma.zona[7]=read_eeprom(ee_alarma_zona7);
alarma.pgm_zona[0]=read_eeprom(ee_alarma_pgm_zona0);
alarma.pgm_zona[1]=read_eeprom(ee_alarma_pgm_zona1);
alarma.pgm_zona[2]=read_eeprom(ee_alarma_pgm_zona2);
alarma.pgm_zona[3]=read_eeprom(ee_alarma_pgm_zona3);
alarma.pgm_zona[4]=read_eeprom(ee_alarma_pgm_zona4);
alarma.pgm_zona[5]=read_eeprom(ee_alarma_pgm_zona5);
alarma.pgm_zona[6]=read_eeprom(ee_alarma_pgm_zona6);
alarma.pgm_zona[7]=read_eeprom(ee_alarma_pgm_zona7);
alarma.disparo_evento_zona[0]=read_eeprom(ee_alarma_disparo_evento_zona0);
alarma.disparo_evento_zona[1]=read_eeprom(ee_alarma_disparo_evento_zona1);
alarma.disparo_evento_zona[2]=read_eeprom(ee_alarma_disparo_evento_zona2);
alarma.disparo_evento_zona[3]=read_eeprom(ee_alarma_disparo_evento_zona3);
alarma.disparo_evento_zona[4]=read_eeprom(ee_alarma_disparo_evento_zona4);
alarma.disparo_evento_zona[5]=read_eeprom(ee_alarma_disparo_evento_zona5);
alarma.disparo_evento_zona[6]=read_eeprom(ee_alarma_disparo_evento_zona6);
alarma.disparo_evento_zona[7]=read_eeprom(ee_alarma_disparo_evento_zona7);
alarma.retardo_zona[0]=read_eeprom(ee_alarma_retardo_zona0);
```

```
  alarma.retardo_zona[1]=read_eeprom(ee_alarma_retardo_zona1);
  alarma.retardo_zona[2]=read_eeprom(ee_alarma_retardo_zona2);
  alarma.retardo_zona[3]=read_eeprom(ee_alarma_retardo_zona3);
  alarma.retardo_zona[4]=read_eeprom(ee_alarma_retardo_zona4);
  alarma.retardo_zona[5]=read_eeprom(ee_alarma_retardo_zona5);
  alarma.retardo_zona[6]=read_eeprom(ee_alarma_retardo_zona6);
  alarma.retardo_zona[7]=read_eeprom(ee_alarma_retardo_zona7);
  alarma.tiempo_pgm[0]=read_eeprom(ee_alarma_tiempo_pgm0);
  alarma.tiempo_pgm[1]=read_eeprom(ee_alarma_tiempo_pgm1);
  alarma.tiempo_pgm[2]=read_eeprom(ee_alarma_tiempo_pgm2);
  alarma.tiempo_pgm[3]=read_eeprom(ee_alarma_tiempo_pgm3);
  n=0;
  do{clave1[n]=read_eeprom(ee_clave1+n);}while(clave1[n++]!='\0');
  n=0;
  do{clave2[n]=read_eeprom(ee_clave2+n);}while(clave2[n++]!='\0');
  n=0;
  do{clave3[n]=read_eeprom(ee_clave3+n);}while(clave3[n++]!='\0');
  n=0;
  do{clave[n]=read_eeprom(ee_clave+n);}while(clave[n++]!='\0');
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(AN0);
  set_adc_channel(0);
  delay_ms(100);
  usb_init_cs();
  enable_interrupts(global);
  enable_interrupts(int_rda);
  rtos_run();
}
```

3.7.3. CONFIGURACIÓN DEL EQUIPO DE MONITOREO.

La configuración del equipo puede efectuarse con cualquier terminal HID el VID y PID del equipo es 1121 y 32 respectivamente. El equipo funciona en dos modos.

El modo monitoreo, este es el modo en el que trabaja por defecto, en este modo el equipo monitorea las entradas, salidas, detecta los eventos, gestiona la conexión al servidor mediante el modem, etc., si el puerto usb está conectado algún terminal HID muestra todos los eventos y mensajes que le llegan.

El modo consola, a este modo solo se puede acceder cuando el equipo está conectado al PC y a un terminal HID, en este modo deshabilita todas las funciones de monitoreo y se puede hacer las configuraciones necesarias.

3.7.4. COMANDOS DE CONFIGURACIÓN.

Para entrar al modo de configuraciones se debe introducir el comando "!cmd consola".

Configurar del apn del proveedor de servicios.

apn=xxx, donde xxx es el nombre del apn del operador, ejemplo.

apn=internet.nuevatel.com

Configuración del IP y puerto del servidor.

ip servidor=xxx.xxx.xxx.xxx,zzz donde xxx es el ip del servidor y zzz es el puerto al que se comunica, ejemplo.

ip servidor=192.168.10.10,5000

Configuración del tiempo de reporte.

reporte=t donde t es el inervalo de tiempo en minutos, ejemplo.

reporte=5 el tiempo puede variar de 1 a 255 segundos

Envío de SMS, este comando determina si se enviara o no un sms al servidor cuando no existan conexión por red.

sms=1 1 envía el SMS, 0 no envía SMs.

Comando para definir el número al que se enviara el sms.

nro sms=xxxx xxxx es el número de móvil al que se enviara el mensaje de texto.

Comando para definir el código de usuario para activar o la alarma.

clavex=zzzz x es el número de clave (1 a 3), zzzz es la clave que pude ser de hasta 8 dígitos.

Comando para habilitar las entradas.

zonax=z x corresponde al número de zona (0 a 7) y z es 0 para deshabilitar la zona, 1 para habilitar la zona y 2 para habilitar la zona en modo pánico.

Comando para asignar una salida al evento de alguna zona.

pgm zonax=z x es la zona (0 a 7) y z es la salida a la que se asigna (0 a 3)

Comando para activar un evento y el modo de disparo.

disparo evento zonax=ae **x** es el número de zona (0 a 7)

a es el estado de la alarma en la que se dispara el evento

0 desactivado, 1 activado, 2 modo presente.

e es el estado en que la entrada dispara el evento

0 entrada abierta, 1 entrada carrada.

Comando para configurar el retardo de zona.

retardo de zonax=t x es el número de zona (0 a 7)

t es el tiempo de retardo en segundos (0 a 255), 0 es evento inmediato.

Comando para ver las configuraciones.

parametros Este comando muestra todos los parámetros configurados incluso los del modem.

Comando para guardar las configuraciones.

guardar Este comando guarda todas las configuraciones.

El equipo también soporta comandos AT para la configuración avanzada del modem.

3.8. TERMINAL HID PARA LA CONFIGURACIÓN DEL EQUIPO DE RASTREO.

Para facilitar un poco la configuración se implementó este terminal en Visual Basic 6 que tiene incorporado los comandos de configuración.

Formulario.

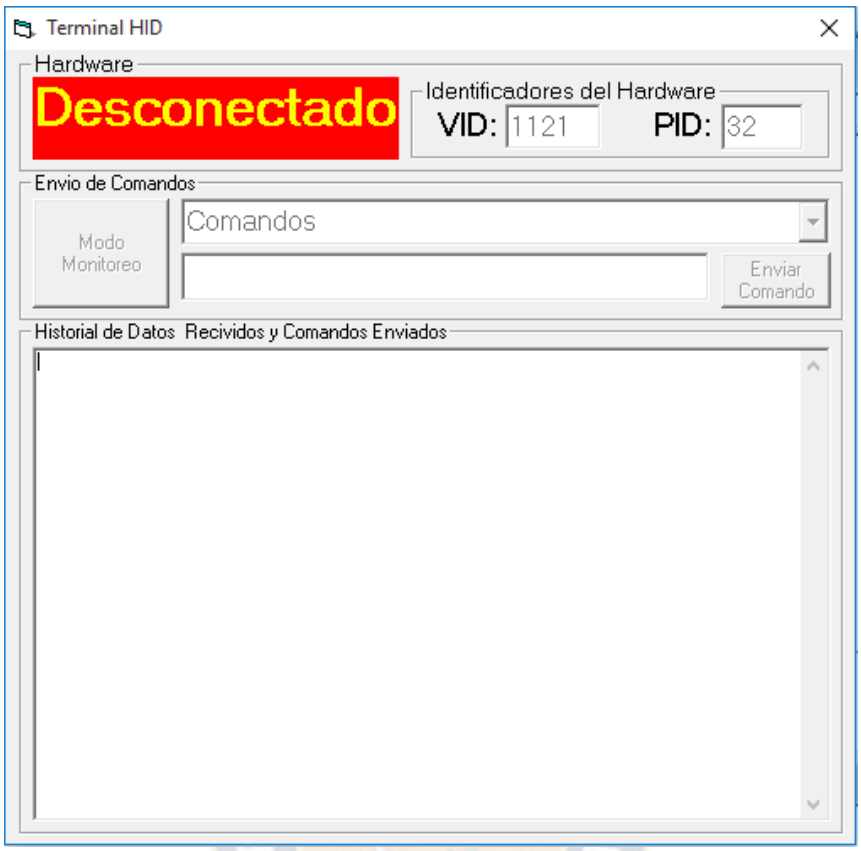


Figura 0.7 Terminal HID para la configuración del equipo de monitoreo.

Programa.

```
' vendor and product IDs
Private Const VendorID = 1121
Private Const ProductID = 32
' read and write buffers
Private Const BufferInSize = 64
Private Const BufferOutSize = 64
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte
Private Sub Combo1_Click()
Text1.Text = Combo1.List(Combo1.ListIndex)
End Sub
Private Sub Command1_Click()
Dim n As Integer
For n = 1 To BufferOutSize Step 1
   If Mid(Text1.Text, n, 1) <> "" Then
      BufferOut(n) = Asc(Mid(Text1.Text, n, 1))
      'Text2.Text = Text2.Text & BufferOut(n)
   Else
      BufferOut(n) = 0
```

```
      'Text2.Text = Text2.Text & BufferOut(n)
    End If
Next
WriteSomeData
End Sub
Private Sub Command2_Click()
Dim cadena As String
If Command2.Caption = "Modo Monitoreo" Then
    Command2.Caption = "Modo Consola"
    Combo1.Enabled = True
    Text1.Enabled = True
    Command1.Enabled = True
    cadena = "!cmd consola"
Else
    Command2.Caption = "Modo Monitoreo"
    Combo1.Enabled = False
    Text1.Enabled = False
    Command1.Enabled = False
    cadena = "!cmd monitoreo"
End If
For n = 1 To BufferOutSize Step 1
  If Mid(cadena, n, 1) <> "" Then
     BufferOut(n) = Asc(Mid(cadena, n, 1))
  Else
     BufferOut(n) = 0
  End If
Next
WriteSomeData
End Sub
' ********************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'*********************************************************************
Private Sub Form_Load()
  ' do not remove!
  ConnectToHID (Me.hwnd)
End Sub
'*********************************************************************
' disconnect from the HID controller...
'*********************************************************************
Private Sub Form_Unload(Cancel As Integer)
  DisconnectFromHID
End Sub
'*********************************************************************
' a HID device has been plugged in...
'*********************************************************************
```

```
Public Sub OnPlugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
    ' ** YOUR CODE HERE **
    Label1.BackColor = &HFF00&
    Label1.ForeColor = &HFF0000
    Label1.Caption = "Conectado"
    Command2.Enabled = True
  End If
End Sub
'*********************************************************************
' a HID device has been unplugged...
'*********************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
    ' ** YOUR CODE HERE **
    Label1.BackColor = &HFF&
    Label1.ForeColor = &HFFFF&
    Label1.Caption = "Desconectado"
    Command2.Enabled = False
    Command2.Caption = "Modo Monitoreo"
    Combo1.Enabled = False
    Text1.Enabled = False
    Command1.Enabled = False
  End If
End Sub
'*********************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*********************************************************************
Public Sub OnChanged()
  Dim DeviceHandle As Long
  ' get the handle of the device we are interested in, then set
  ' its read notify flag to true - this ensures you get a read
  ' notification message when there is some data to read...
  DeviceHandle = hidGetHandle(VendorID, ProductID)
  hidSetReadNotify DeviceHandle, True
End Sub
'*********************************************************************
' on read event...
'*********************************************************************
Public Sub OnRead(ByVal pHandle As Long)
  Dim n As Byte
  Dim rx_txt As String
  ' read the data (don't forget, pass the whole array)...
  If hidRead(pHandle, BufferIn(0)) Then
    ' ** YOUR CODE HERE **
    ' first byte is the report ID, e.g. BufferIn(0)
```

```
    ' the other bytes are the data from the microcontrolller...
    For n = 1 To BufferInSize Step 1
      If BufferIn(n) = 0 Then
         Exit For
      End If
      rx_txt = rx_txt & Chr(BufferIn(n))
    Next
    Text2.Text = Text2.Text & rx_txt & vbCrLf
    Text2.SelStart = Len(Text2.Text)
  End If
End Sub
'*********************************************************************
' this is how you write some data...
'*********************************************************************
Public Sub WriteSomeData()
  BufferOut(0) = 0   ' first by is always the report ID
  'BufferOut(1) = 10  ' first data item, etc etc

  ' write the data (don't forget, pass the whole array)...
  hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub
```

Modulo adicionado para el manejo del archivo *mcHID.dll*

```
' this is the interface to the HID controller DLL - you should not
' normally need to change anything in this file.
'
' WinProc() calls your main form 'event' procedures - these are currently
' set to..
'
' MainForm.OnPlugged(ByVal pHandle as long)
' MainForm.OnUnplugged(ByVal pHandle as long)
' MainForm.OnChanged()
' MainForm.OnRead(ByVal pHandle as long)

Option Explicit

' HID interface API declarations...
Declare Function hidConnect Lib "mcHID.dll" Alias "Connect" (ByVal pHostWin As Long) As
Boolean
Declare Function hidDisconnect Lib "mcHID.dll" Alias "Disconnect" () As Boolean
Declare Function hidGetItem Lib "mcHID.dll" Alias "GetItem" (ByVal pIndex As Long) As Long
Declare Function hidGetItemCount Lib "mcHID.dll" Alias "GetItemCount" () As Long
Declare Function hidRead Lib "mcHID.dll" Alias "Read" (ByVal pHandle As Long, ByRef pData
As Byte) As Boolean
Declare Function hidWrite Lib "mcHID.dll" Alias "Write" (ByVal pHandle As Long, ByRef
pData As Byte) As Boolean
```

Declare Function hidReadEx Lib "mcHID.dll" Alias "ReadEx" (ByVal pVendorID As Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean
Declare Function hidWriteEx Lib "mcHID.dll" Alias "WriteEx" (ByVal pVendorID As Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean
Declare Function hidGetHandle Lib "mcHID.dll" Alias "GetHandle" (ByVal pVendoID As Long, ByVal pProductID As Long) As Long
Declare Function hidGetVendorID Lib "mcHID.dll" Alias "GetVendorID" (ByVal pHandle As Long) As Long
Declare Function hidGetProductID Lib "mcHID.dll" Alias "GetProductID" (ByVal pHandle As Long) As Long
Declare Function hidGetVersion Lib "mcHID.dll" Alias "GetVersion" (ByVal pHandle As Long) As Long
Declare Function hidGetVendorName Lib "mcHID.dll" Alias "GetVendorName" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetProductName Lib "mcHID.dll" Alias "GetProductName" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetSerialNumber Lib "mcHID.dll" Alias "GetSerialNumber" (ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetInputReportLength Lib "mcHID.dll" Alias "GetInputReportLength" (ByVal pHandle As Long) As Long
Declare Function hidGetOutputReportLength Lib "mcHID.dll" Alias "GetOutputReportLength" (ByVal pHandle As Long) As Long
Declare Sub hidSetReadNotify Lib "mcHID.dll" Alias "SetReadNotify" (ByVal pHandle As Long, ByVal pValue As Boolean)
Declare Function hidIsReadNotifyEnabled Lib "mcHID.dll" Alias "IsReadNotifyEnabled" (ByVal pHandle As Long) As Boolean
Declare Function hidIsAvailable Lib "mcHID.dll" Alias "IsAvailable" (ByVal pVendorID As Long, ByVal pProductID As Long) As Boolean

' windows API declarations - used to set up messaging...
Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

' windows API Constants
Private Const WM_APP = 32768
Private Const GWL_WNDPROC = -4

' HID message constants
Private Const WM_HID_EVENT = WM_APP + 200
Private Const NOTIFY_PLUGGED = 1
Private Const NOTIFY_UNPLUGGED = 2
Private Const NOTIFY_CHANGED = 3
Private Const NOTIFY_READ = 4

```
' local variables
Private FPrevWinProc As Long    ' Handle to previous window procedure
Private FWinHandle As Long      ' Handle to message window

' Set up a windows hook to receive notification
' messages from the HID controller DLL - then connect
' to the controller
Public Function ConnectToHID(ByVal pHostWin As Long) As Boolean
  FWinHandle = pHostWin
  ConnectToHID = hidConnect(FWinHandle)
  FPrevWinProc = SetWindowLong(FWinHandle, GWL_WNDPROC, AddressOf WinProc)
End Function

' Unhook from the HID controller and disconnect...
Public Function DisconnectFromHID() As Boolean
  DisconnectFromHID = hidDisconnect
  SetWindowLong FWinHandle, GWL_WNDPROC, FPrevWinProc
End Function

' This is the procedure that intercepts the HID controller messages...
Private Function WinProc(ByVal pHWnd As Long, ByVal pMsg As Long, ByVal wParam As
Long, ByVal lParam As Long) As Long
  If pMsg = WM_HID_EVENT Then
    Select Case wParam

      ' HID device has been plugged message...
      Case Is = NOTIFY_PLUGGED
        MainForm.OnPlugged (lParam)

      ' HID device has been unplugged
      Case Is = NOTIFY_UNPLUGGED
        MainForm.OnUnplugged (lParam)

      ' controller has changed...
      Case Is = NOTIFY_CHANGED
        MainForm.OnChanged

      ' read event...
      Case Is = NOTIFY_READ
        MainForm.OnRead (lParam)
    End Select
  End If

  ' next...
  WinProc = CallWindowProc(FPrevWinProc, pHWnd, pMsg, wParam, lParam)

End Function
```

CAPITULO IV

COSTOS Y PRESUPUESTOS.

4.1. COSTOS DEL CIRCUITO DE MONITOREO.

Category	Quantity	References	Value	Unit Cost (Bs)
Capacitors	1	C1	2200uF	1,50
Capacitors	1	C2	220uF	1,00
Capacitors	1	C3	270pF	0,50
Capacitors	1	C7	100nF	0,50
Capacitors	2	C8,C9	22pF	0,50
Resistors	1	R1	20k	0,20
Resistors	1	R2	51k	0,20
Resistors	1	R3	0.56	0,20
Resistors	4	R4,R16,R17,R18	330	0,20
Resistors	1	R5	2	0,20
Resistors	1	R6	120	0,20
Resistors	2	R7,R19	10k	0,20
Resistors	8	R8,R9,R10,R11,R12,R13,R14,R15	1k	0,20
Integrated Circuits	1	U1	MC34063	5,00
Integrated Circuits	1	U2	PIC18F4550	60,00
Integrated Circuits	1	U3	ULN2803	6,00
Integrated Circuits	1	U4	4051	3,50
Integrated Circuits	1	U5	MAX485	20,00
Diodes	2	D1,D6	1N4007	0,50
Diodes	2	D2,D3	1N5404	1,50
Diodes	1	D4	1N4819	1,00
Diodes	4	D5,D7,D8,D9	LED-RED	0,50
Miscellaneous	3	J1,J2,J3	BORNERA2	3,00
Miscellaneous	8	J4,J5,J6,J7,J10,J11,J12,J13	BORNERA3	3,50
Miscellaneous	1	J8	CONECTOR USB	3,00
Miscellaneous	1	J9	MODEM GSM	100,00
Miscellaneous	1	J14	BUTTON	1,00
Miscellaneous	2	J15,J16	ESPADIN	6,00
Miscellaneous	1	L1	2.7mH	5,00
Miscellaneous	1	RJ1	RJ45	5,00
Miscellaneous	4	RL1,RL2,RL3,RL4	G5CLE-1-DC12	6,00
Miscellaneous	1	X1	CRYSTAL	4,00

Totales

Category	Quantity	Unit Cost (Bs).
Capacitors	6	4,5
Resistors	19	3,8
Integrated Circuits	5	94,5
Diodes	9	7
Miscellaneous	23	191
Total	**62**	**300,8**

Costos del teclado.

Category	Quantity	References	Value	Unit Cost(Bs)
Capacitors	1	C1	220p	0,50
Capacitors	1	C2	220U	1,00
Capacitors	1	C3	100n	0,50
Resistors	1	R1	100k	0,20
Resistors	1	R2	0.62	0,20
Resistors	1	R3	10k	0,20
Resistors	1	R4	30k	0,20
Resistors	1	R5	120	0,20
Integrated Circuits	1	U1	PIC16F876A	40,00
Integrated Circuits	1	U2	MAX485	20,00
Integrated Circuits	1	U4	MC34063	5,00
Diodes	1	D1	1N4819	1,50
Miscellaneous	1	BUZ1	BUZZER	5,00
Miscellaneous	1	K1	KEYPAD1	35,00
Miscellaneous	1	L1	2.7mH	5,00
Miscellaneous	1	LCD1	LM016L	50,00
Miscellaneous	1	RJ1	RJ45	5,00
Miscellaneous	1	RV1	PRESET	1,50
Miscellaneous	1	X1	CRYSTAL	4,00

Totales

Category	Quantity	Unit Cost(Bs)
Capacitors	3	2
Resistors	5	1
Integrated Circuits	3	65
Diodes	1	1,5
Miscellaneous	7	105,5
Total	**19**	**175**

CONCLUSIONES Y RECOMENDACIONES.

CONCLUSIONES.

En el presente proyecto afortunadamente se logró cumplir todos los objetivos trazados tanto en software como hardware.

Con el modem GSM se tuvieron algunas dificultades al momento de adquirir y montarlo en la placa de circuito impreso, fuera de ese detalle no existió mayor inconveniente.

Las pruebas realizadas del funcionamiento del proyecto fueron satisfactorias gracias a que en la facultad se contaba con servicio de internet y disponibilidad de IP público ya que de no ser así sería muy difícil hacer la demostración práctica.

El tiempo transcurrido entre la generación del evento (disparo de algún sensor) hará la llegada del reporte al servidor fue de 2 segundos como valor máximo.

RECOMENDACIONES.

Para que este sistema de supervisión remota no tenga inconvenientes en su funcionamiento es recomendable que en la zona donde se instale el equipo de monitoreo exista un servicio aceptable de internet móvil por parte de alguna operadora de telefonía móvil.

Para darle más prestaciones a este sistema de supervisión se podría utilizar un modem 4G y así poder utilizar un mayor ancho de banda para enviar fotografías (de una cámara de video digital para sistemas embebidos) cuando se genere un evento de alarma.

Para un proyecto futuro se podría adicionar la posibilidad de efectuar las configuraciones del equipo mediante SMS o en línea con el servidor, así también poder actualizar el firmware del equipo de manera remota

BIBLIOGRAFÍA

(s.f.). Obtenido de php.net: http://php.net/manual/es/faq.general.php

(s.f.). Obtenido de DesarrolloWeb.com: http://www.desarrolloweb.com/php/

(s.f.). Obtenido de CCS: http://www.ccsinfo.com/content.php?page=compilers

(2016). Obtenido de Futurlec: http://www.futurlec.com/Microchip/PIC18F4550.shtml

Alberkrack, J. (abril de 2002). Obtenido de ON Semiconductor: http://onsemi.com

Alvarez, M. A. (14 de marzo de 2002). Obtenido de desarrolloweb.com: http://www.desarrolloweb.com/articulos/711.php

Alvarez, M. A. (18 de julio de 2007). Obtenido de desarrolloweb.com: http://www.desarrolloweb.com/articulos/xampp.html

Apache Friends. (2016). Obtenido de Apache Friends: https://www.apachefriends.org/index.html

Casares, C. (s.f.). Obtenido de Tutoriales de la Cueva: http.//usuarios.tripod.es/smaug

Casares, C. (07 de septiembre de 2004). Obtenido de Maestros del web: http://www.maestrosdelweb.com/editorial/tutsql1/

Criado, A. B. (04 de febrero de 2008). Obtenido de adictosaltrabajo.com: https://www.adictosaltrabajo.com/tutoriales/xampp/

Garcia de Jalon, J., Rodriguez, J. I., & Brazalez, A. (1999). *Aprenda Visual Basic 6.0.* San Sebastian: Universidad de Navarra.

Gemu. (septiembre de 2005). Obtenido de monografias.com: http://www.monografias.com/trabajos30/tutorial-visual-basic/tutorial-visual-basic.shtml

Inc., 2. M. (s.f.). Obtenido de http://ww1.microchip.com/downloads/en/devicedoc/39632e.pdf

Lapuente, M. J. (08 de diciembre de 2013). Obtenido de hipertexto.info: http://www.hipertexto.info/documentos/html.htm

leigh, k. (s.f.). Obtenido de eHow: http://www.ehowenespanol.com/funciona-sensor-rotura-vidrios-como_117342/

Mejía, M. (s.f.). Obtenido de Seguridad Primero: https://seguridadprimero.wordpress.com/configurando-su-sistema-de-seguridad/parte-5-deteccion-de-ruptura-de-vidrios/

Morales, A. (29 de septiembre de 2010). Obtenido de http://www.ingeniero.moralesm.name/2010/09/29/clase-request-2/

Ojeda, L. T. (abril de 2010). Obtenido de olimex.cl: http://www.olimex.cl/tutorial/MCI-Guia%20de%20usuario%20Enfora%20v2.pdf

Quesada, J. L. (julio de 2008). Obtenido de Universidad de Costa Rica Facultad de Ingeniería Escuela de Ingeniería Eléctrica: http://eie.ucr.ac.cr/uploads/file/proybach/pb0819t.pdf

Ravioli, P. (s.f.). Obtenido de Monografias.com: http://www.monografias.com/trabajos7/html/html.shtml

Serrano, C. Z. (05 de abril de 2013). Obtenido de EL BLOG ELECTRONICO DE CZZS: http://czzsjd2.blogspot.com/2013/04/estructura-interna-del-pic-18f4550.html

Villegas, J. (02 de febrero de 2012). Obtenido de TECNOSeguro.com: https://www.tecnoseguro.com/faqs/alarma/que-es-un-detector-de-movimiento-pasivo-o-pir.html