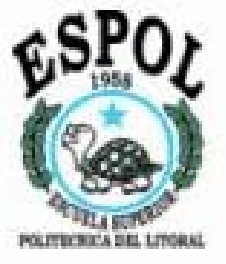

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Administración de imágenes de ecografía para generar informes médicos para usos de diagnósticos"

TOPICO DE GRADO

Previa a la obtención del título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA

Presentada por:

Braulio Eduardo Cerón B.
Miguel Angel Sandoval R.

GUAYAQUIL – ECUADOR
2003

AGRADECIMIENTO

El autor agradece al Ing. Miguel Yapur, Director de Tópico, quien con su valiosa orientación, conocimiento e incentivo me ayudó a desarrollar este trabajo.

A todas las personas que de una u otra manera colaboraron para la realización de este trabajo.

DEDICATORIA

A Dios y a mis queridos padres, hermanos y familiares, quienes en todo momento me brindan su apoyo para vencer los obstáculos.

TRIBUNAL DE GRADUACIÓN

.....	.....
Ing. Norman Chootong	Ing. Miguel Yapur A.
SUBDECANO DE LA FIEC	DIRECTOR DE TOPICO
.....	.....
Ing. Juan del Pozo L.	Ing. Hugo Villavicencio
VOCAL PRINCIPAL	VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

.....

Braulio Eduardo Cerón B.

.....

Miguel Ángel Sandoval R.

RESUMEN

El proyecto consiste en elaborar una interfase gráfica de usuario usando el lenguaje de Microsoft Visual Basic 6.0 con la finalidad de mostrar mediante el enlace de audio y video entrelazado de un ecógrafo con la entrada de una tarjeta capturadora de video acoplada a un computador personal para que la administración post diagnóstico de los informes sea más efectivo.

Esta interfase gráfica permite al usuario obtener fotos o secciones de video en formato JPG y AVI respectivamente para después almacenarlas junto con los datos personales del paciente y formar una base de datos.

La interfase gráfica de usuario da la posibilidad de acceder ya sea directa e indirectamente a la base de datos con información personal junto con las imágenes de ecografía más representativas del diagnóstico del paciente para luego imprimir en una simple hoja de papel con una buena resolución de imagen.

El desarrollo y aplicación de este trabajo se basó en el diseño e implementación de una interfase ecógrafo-computadora por medio de una tarjeta capturadora de video realizado por estudiantes de un anterior Tópico de Graduación de la Facultad.

INDICE GENERAL

Pág.

INDICE DE FIGURAS

Pág.

CAPITULO I

CAPITULO IV

ANEXO DOS

INTRODUCCION

En el campo médico las imágenes se han convertido en ayuda importante para los profesionales; se utilizan para realizar diagnósticos, planear una forma de tratamiento y monitorear cambios en el tiempo de alguna patología. Estas aplicaciones, por cierto especializadas, requieren estandarización en la forma como se deben obtener, procesar, presentar y almacenar las imágenes, con el fin de asegurar la confiabilidad de interpretación durante el procedimiento que se realice.

Existen inconvenientes por calidad de la imagen, se presentan problemas en cuanto almacenamiento, manipulación y accesibilidad de las mismas, lo que se ve reflejado en dificultad para la realización de seguimiento a mediano y largo plazo de pacientes, inadecuada evaluación de tratamientos, lentitud en la consulta de expedientes y pérdida de información, entre otros, los cuales disminuyen el nivel de calidad de los servicios médicos prestados.

La idea del mejoramiento, en general, ha sido el poder dar una mejor administración de la imagen original; lo cual ayuda a una rápida búsqueda, visualización y por lo tanto una mejor interpretación o diagnostico.

Para la parte de almacenamiento y manipulación de las imágenes se presenta el diseño de un sistema de información enfocado a consultas en obstetricia, la implementación de este sistema de información y la fusión de los dos enfoques mediante un sistema que integra la resolución de los problemas mencionados.

CAPITULO UNO

DESARROLLO DEL PROGRAMA

Para el desarrollo del proyecto, se debe poseer un mínimo requerimiento de hardware, además del conocimiento del programa Microsoft Visual Basic 6.0 .

Se seleccionó Microsoft Visual Basic 6.0 porque actualmente es el lenguaje de programación más popular del mundo diseñado especialmente para crear aplicaciones gráficas de manera fácil y eficiente, tal es el caso que para diseñar la interfase gráfica de usuario, sólo es necesario “dibujar” ciertos objetos prefabricados llamados controles en un formulario dentro de la pantalla para luego escribir el código asociado con cada objeto.

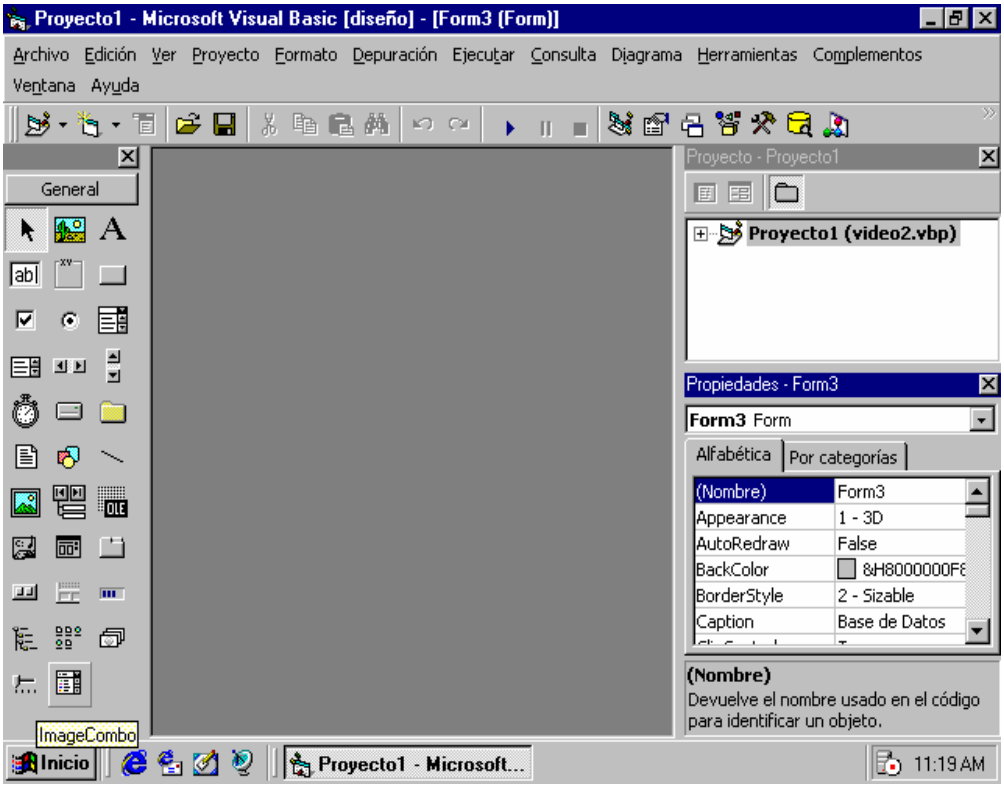

Fig.1.1. Entorno de programación en Microsoft Visual Basic

En conjunción a este lenguaje de programación se seleccionó para la base de datos un archivos aleatorio conocido también como archivo de acceso directo. Los pasos a seguir en la manipulación de archivos aleatorios son muy sencillos utilizando comandos de Microsoft Visual Basic 6.0.

En el ambiente de Microsoft Visual Basic 6.0 se desarrollarán tres formularios, la interfase gráfica de usuario principal, el formulario de posición para impresión y la base de datos con los controles y propiedades respectivamente.

1.1. REQUERIMIENTOS DE HARDWARE

Para la realización del proyecto se requiere un mínimo de requisitos de cumplimiento en hardware, esto es:

- Una computadora con procesador PENTIUM de 90 Mhz mínimo recomendado
- Windows 95, 98 o NT 4.0
- Slots PCI libres.
- Tarjeta de sonido.
- CD rom drive.
- Tarjeta VGA(PCI o AGP)
- Ecógrafo con salida de video compuesto.

1.2. DISEÑO DE LA INTERFASE GRAFICA DE USUARIO

La interfase gráfica de usuario es el medio de comunicación entre la aplicación dada en el computador y el usuario.

El diseño de la interfase de usuario de una aplicación consiste en “dibujar” los controles sobre el formulario. Para ello, se debe utilizar el cuadro de herramientas.

Se utilizara los ficheros AVI cuya característica principal es utilizarse con aplicaciones que capturan, editan y reproducen películas de audio y video entrelazado. Para tomar el control sobre el reproductor de AVI asigne a la propiedad DeviceType el valor “AVIVideo”.

A continuación se indican los controles que se utilizaran para el diseño de la interfase gráfica de usuario:

Tabla I. Controladores para el diseño de la interfase gráfica de usuario

Cantidad	**Controles**
3	Formularios
19	Botón de comando
1	Menú
2	Caja de imagen
2	Control MCI
1	Cuadro de lista de unidades
1	Cuadro de lista de directorio
1	Cuadro de lista de archivos
6	Marco
1	Cuadro de dialogo estándar
1	Caja de lista
14	Caja de texto (Arreglo de controles 0,1,2,...,13)
9	Etiquetas

1.3. PROPIEDADES DE LOS CONTROLES

Luego de diseñar la interfase de usuario, el siguiente paso consiste en establecer las propiedades de los elementos incluidos en el formulario.

Para ello es necesario utilizar la ventana **Propiedades**. Si esta ventana no esta visible, hacemos clic en el botón **Ventana Propiedades** o presione la tecla **F4**.

Por defecto, Visual Basic asigna un nombre a cada uno de los controles dibujados en el formulario y al formulario mismo. El nombre de un control nos permite asociarlo con un procedimiento y referirnos a él dentro del código de un programa para modificar algunas de sus propiedades.

Modifiquemos y establezcamos las principales propiedades a los controles de la interfase gráfica de usuario principal:

Tabla II. Propiedades de los controladores de la interfase gráfica de Form1.

Controlador	**Propiedad**	**Valor**
Formulario	Caption	Enlace Ecógrafo - PC
	Name	Form1
CommandButton	Caption	JPG
	Name	JPG
	Picture	Icono
CommandButton	Caption	AVI
	Name	AVI
	Picture	Icono
CommandButton	Caption	Base de Datos

	Name	BasedeDatos
	Picture	Icono
CommandButton	Caption	Imprimir
	Name	Impresión
	Picture	Mapa de bit
Frame	Caption	Unidad:
	Name	Frame2
Frame	Caption	Directorios:
	Name	Frame3
Frame	Caption	Frame4
	Name	Archivos:
Frame	Caption	Frame1
	Name	Ecograma
Caja de Imagen	Name	Picture1
	Picture	(ninguno)
Control MCI	Name	MMControl1
	Eject Visible	False
	Record Visible	False
DriveListBox	Name	Drive1
DirListBox	Name	Dir1
FileListBox	Name	File1
Common Dialog	Name	CommonDialog1
Menú	Caption	&Abrir
	Name	FicheroAbrir
Menú	Caption	&Fichero
	Name	MenúFichero
Menú	Caption	&Imprimir
	Name	Imprimir
Menú	Caption	&Salir
	Name	FicheroSalir
Menú	Caption	&Opciones
	Name	Opciones
Menú	Caption	&AVI sonido
	Name	AVISonido
Menú	Caption	&Sin sonido
	Name	AVISinsonido
Menú	Caption	&Con sonido
	Name	AVIConsonido

Definidas todas las propiedades de los controladores de la interfase gráfica de usuario principal, se muestra el entorno en la siguiente figura

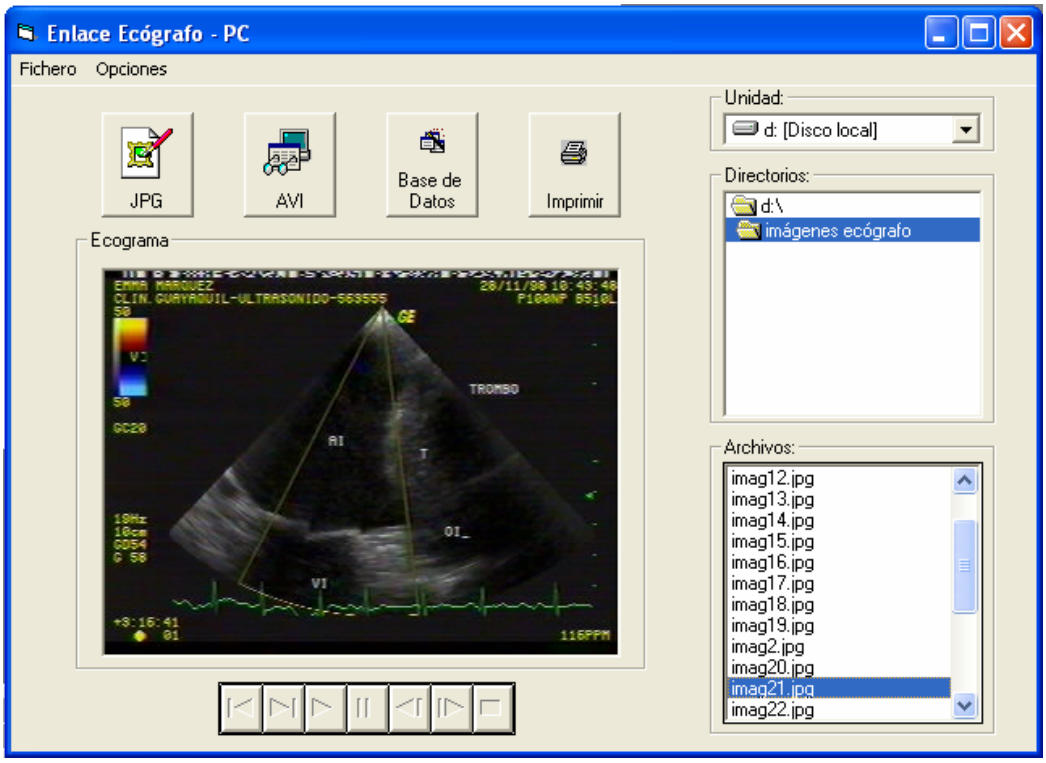

Fig. 1.2. Interfase gráfica del formulario principal

Se definen las propiedades de los controladores de la segunda interfase gráfica de usuario:

Tabla III. Propiedades de los controladores de la interfase gráfica de Form2

Controlador	**Propiedad**	**Valor**
Formulario	Caption	Form2
	Name	Posición
Frame	Caption	Frame1
	Name	Hoja
Frame	Caption	Frame2
	Name	(ninguna)
CommandButton	Caption	&Cancelar

	Name	Cancelar
CommandButton	Caption	Command1
	Name	1
CommandButton	Caption Name	Command2 2
CommandButton	Caption	Command3
	Name	3
CommandButton	Caption	Command4
	Name	4
CommandButton	Caption Name	Command5 5
CommandButton	Caption	Command6
	Name	6
CommandButton	Caption	Command7
	Name	7
CommandButton	Caption	Command8
	Name	8

Definidas las propiedades del formulario de impresión, entonces quedará de la siguiente manera

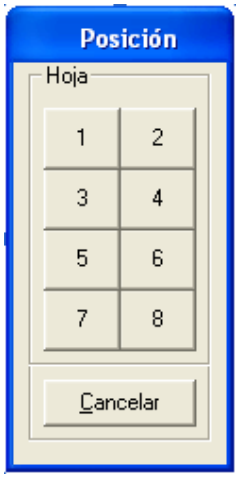

Fig. 1.3. Interfase gráfica del formulario de impresión

Tabla IV. Propiedades de los controladores de la interfase gráfica de MantDatos

Controlador	**Propiedad**	**Valor**
Formulario	Caption	Base de Datos de pacientes

	Name	MantDatos
ListBox	Name	lstCódigo
Caja de Imagen	Name	Picture1
	Picture	(ninguno)
Control MCI	Name	MMControl1
	Eject Visible	False
	Record Visible	False
CommandButton	Caption	&Eliminar
	Name	cmdEliminar
CommandButton	Caption	Guardar
	Name	cmdGuardar
CommandButton	Caption	Ingresar
	Name	cmdIngresar
CommandButton	Caption	Modificar
	Name	cmdModificar
CommandButton	Caption	&Salir
	Name	cmdSalir
CommandButton	Caption	&Salir
	Name	cmdSalir
Label	Caption	Imprimir
	Name	Impresión
Label	Caption	Número de cédula:
	Name	Label2
Label	Caption	Nombre:
	Name	Label3
Label	Caption	Dirección
	Name	Label4
Label	Caption	Teléfono:
	Name	Label5
Label	Caption	Parte Cuerpo:
	Name	Label6
Label	Caption	AVI:
	Name	Label7
Label	Caption	JPG:
	Name	Label8
Label	Caption	Diagnóstico:
	Name	Label9
Array de Controles TextBox	Name	txtCampo
	Text	“ “
	Index	0,1,2,...,13

Definidas las propiedades de los controladores para la base de datos, debe quedar como muestra la figura.

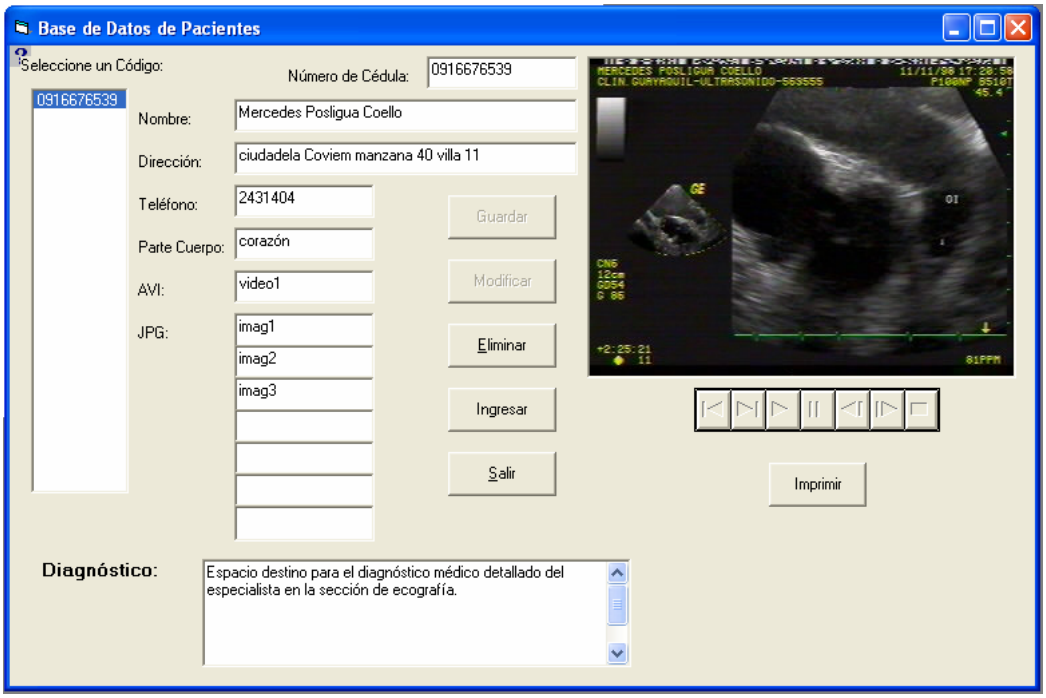

Fig. 1.4. Interfase gráfica del formulario de la base de datos

CAPITULO DOS

PROGRAMACIÓN

2.1. CODIGO ASOCIADO A FORM1 (FORMULARIO PRINCIPAL)

Añadimos al formulario un control de diálogo estándar CommonDialog con el fin de obtener el nombre del fichero .avi que deseamos reproducir y una caja de imagen Picture sobre la que se reproducirá dicho fichero. Para visualizar la caja de diálogo estándar y cerrar la aplicación añadimos también al formulario el menú Fichero con las órdenes Abrir, Salir e Imprimir y el menú Opciones con la orden AVI Sonido.

La orden Imprimir accionada por el suceso **Click** ejecuta el procedimiento Impresión

```
Private Sub Imprimir_Click()
  Impresion_Click
End Sub
```

La orden Salir del menú Fichero simplemente descarga el formulario (se produce el suceso **Unload**) y finaliza la aplicación. Aprovechamos el procedimiento conducido por el suceso **Unload** para cerrar el control MCI, ya que este suceso se produce también cuando cerramos la aplicación a través de la orden Cerrar del menú de control del formulario.

```
Private Sub FicheroSalir_Click()
  Unload Me
  End
End Sub
Private Sub Form_unLoad(Cancel As Integer)
  MMControl1.Command = "Close"
End Sub
```

A continuación, el procedimiento FicheroAbrir_Click visualiza la caja de diálogo estándar Abrir, asigna el nombre del fichero JPG o AVI elegido a la propiedad **Filename** del control MCI y abre dicho control para que pueda reproducir el fichero seleccionado.

```
Private Sub FicheroAbrir_Click()
  Dim F As String
  ' Propiedades de la caja de diágo Abrir
  CommonDialog1.Filter = "Ficheros (*.jpg;*.avi)|*.jpg;*.avi"
  CommonDialog1.FilterIndex = 1
  ' Visualizar la caja de diágogo Abrir
  CommonDialog1.ShowOpen
  ' Nombre del fichero elegido
  F = CommonDialog1.FileTitle
  MMControl1.FileName = F
  If File1.Pattern = "*.jpg" Then
    ' Abrir imagen JPG
    Picture1.Picture = LoadPicture(F)
  Else
    ' Abrir el dispositivo MCI de audio-video (avi)
    MMControl1.Command = "Open"
  End If
End Sub
```

Cuando el usuario ejecute la aplicación, lo primero que se hace es estableces las propiedades requeridas para abrir el dispositivo MCI. Este proceso se realiza desde el procedimiento **Form_Load.**

```
Private Sub Form_Load()
```

```
Move (Screen.Width - Width) \ 2, (Screen.Height - Height) \ 2
  ' Establecer las propiedades requeridas para abrir MCI
  MMControl1.Notify = True ' para Done
  MMControl1.Wait = True
  MMControl1.Shareable = False
  MMControl1.DeviceType = "AVIVideo"
  MMControl1.hWndDisplay = Picture1.hWnd
End Sub
```

El paso siguiente es seleccionar el fichero que se desea reproducir. Para sincronizar los controles Drive1, Dir1 y File1 piensen en que el usuario primero seleccionará una unidad de disco (control Drive1). Un clic sobre este control hace que este reconozca el suceso **Change**. Por lo tanto, para que se visualice simultáneamente la lista de directorios correspondiente a la nueva unidad seleccionada, el procedimiento Drive1_Change tiene que asignar la nueva unidad a la propiedad Path del control Dir1:

```
Private Sub Drive1_Change()
  ' Path = directorio actual en la unidad especificada por Drive
  Dir1.Path = Drive1.Drive
End Sub
```

El control Dir1 visualiza la nueva lista de directorios y reconoce el suceso **Change**, lo que hace que se ejecute el proceso Dir1_Change. Para que se visualice simultáneamente la lista de ficheros correspondiente al directorio actual con la extensión especificada por la propiedad **Pattern** del control File1 (Form_Load establece esta extensión por defecto) el procedimiento Dir1_Change tiene que asignar el nuevo camino a la propiedad **Path** del control File1:

```
Public Sub Dir1_Change()
  File1.Path = Dir1.Path
End Sub
```

El control File1 visualiza la nueva lista de ficheros. Para seleccionar uno, el usuario hará clic sobre él y para reproducirlo, hará clic en el botón Play. Esto quiere decir que el procedimiento File1_Click debe obtener la ruta de acceso al fichero elegido y asignar la propiedad **picture** de la caja de imagen si es un archivo es JPG o la propiedad **filename** del control MCI si es AVI, dicho fichero para así abrirlo cuando se ejecute la orden **MCI open**.

```
Private Sub File1_Click()
  Dim camino As String
  MMControl1.Command = "Close"
```

```
  If Right(Dir1.Path, 1) = "\" Then
    camino = Dir1.Path + File1.FileName
  Else
    camino = Dir1.Path + "\" + File1.FileName
  End If
  If File1.Pattern = "*.jpg" Then
    Picture1.Picture = LoadPicture(camino)
  Else
    MMControl1.FileName = camino
    MMControl1.Command = "Open"
  End If
End Sub
```

La función del botón JPG, AVI es cerrar el dispositivo MCI abierto, elegir el tipo de fichero .jpg o .avi, respectivamente. Los procedimientos asociados se escriben a continuación:

```
Private Sub AVI_Click()
  MMControl1.Command = "Close"
  File1.Pattern = "*.avi"
End Sub

Private Sub JPG_Click()
```

```
  MMControl1.Command = "Close"
  File1.Pattern = "*.jpg"
End Sub
```

Con la opción AVI sin sonido lo que deseamos es desconectar el sonido durante la reproducción de un AVI (esta opción no funciona con el controlador MCI para ficheros MIDI y WaveAudio). Para ello lo único que tenemos que hacer es asignar a la propiedad **Silent** del control MCI el valor **True** (si sonido) cuando el usuario active la opción y cuando la desactive el valor **False** (con sonido).

```
Private Sub AVIConsonido_Click()
  If MMControl1.Command = "Open" Then
    MMControl1.Silent = False ' sonido desconectado
    AVIConsonido.Enabled = False
    AVISinsonido.Enabled = True
  End If
End Sub

Private Sub AVISinsonido_Click()
  If MMControl1.Command = "Open" Then
    MMControl1.Silent = True ' sonido conectado
    AVISinsonido.Enabled = False
```

```
    AVIConsonido.Enabled = True
  End If
End Sub
```

La caja de imagen Picture1 la hacemos pública para poder utilizarla en el formulario de impresión.

```
Public Sub Picture1_Click()
End Sub
```

Los controles CommandButton de Impresión y Base de Datos están asociados por el suceso **Click** a procedimientos que descargan cada uno su formulario (se produce el suceso **Show**).

```
Private Sub BasedeDatos_Click()
  MantDatos.Show
End Sub

Private Sub Impresion_Click()
  If File1.Pattern = "*.jpg" Then
    Form2.Show
  End If
```

```
End Sub
```

2.2. CODIGO ASOCIADO A FORM2 (FORMULARIO DE IMPRESIÓN)

Cuando activamos el formulario de impresión se presenta la posición de la imagen en la hoja donde deseamos imprimir.

Primero definimos variables correspondientes a las coordenadas **x** y **y** de la zona de recorte dentro de la imagen donde **x** representa la anchura y **y** la altura

```
Dim escalaX As Integer
Dim escalaY As Integer
```

Las dimensiones del papel utilizado las obtenemos utilizando las propiedades **Height** y **Width;** estas dimensiones vienen expresadas en twips.

```
Private Sub Form_Load()
  escalaX = Printer.Width
  escalaY = Printer.Height
End Sub
```

El botón Cancelar abortar la impresión cancelando primero la operación y después cerrando el formulario.

```
Private Sub Cancelar_Click()
  ' Abortar la operación de impresión
  Printer.KillDoc
  Unload Me
End Sub
```

PaintPicture es un nuevo método incluido a partir de la versión 4 de Visual Basic. Este método presenta el contenido de un fichero gráfico (.BMP,.WMF,.EMF,.ICO,.DIB,.JPG).

Los códigos asociados a la posición del gráfico en la hoja se presentan a continuación:

```
Private Sub Command1_Click()
  Printer.PaintPicture Form1.Picture1.Picture, escalaX / 15, escalaY / 28
  Finimpresión
End Sub
```

```
Private Sub Command2_Click()
  Printer.PaintPicture  Form1.Picture1.Picture,  escalaX  /  2  +  escalaX  /  40,
escalaY / 28
  Finimpresión
End Sub

Private Sub Command3_Click()
  Printer.PaintPicture  Form1.Picture1.Picture,  escalaX  /  15,  escalaY  /  4  +
escalaY / 56
  Finimpresión
End Sub

Private Sub Command4_Click()
  Printer.PaintPicture  Form1.Picture1.Picture,  escalaX  /  2  +  escalaX  /  40,
escalaY / 4 + escalaY / 56
  Finimpresión
End Sub

Private Sub Command5_Click()
  Printer.PaintPicture Form1.Picture1.Picture, escalaX / 15, escalaY / 2
  Finimpresión
End Sub
Private Sub Command6_Click()
```

```
 Printer.PaintPicture Form1.Picture1.Picture, escalaX / 2 + escalaX / 40,
escalaY / 2
 Finimpresión
End Sub

Private Sub Command7_Click()
 Printer.PaintPicture Form1.Picture1.Picture, escalaX / 15, escalaY / 2 +
escalaY / 4 - escalaY / 56
 Finimpresión
End Sub

Private Sub Command8_Click()
 Printer.PaintPicture Form1.Picture1.Picture, escalaX / 2 + escalaX / 40,
escalaY / 2 + escalaY / 4 - escalaY / 56
 Finimpresión
End Sub
```

2.3. CÓDIGO ASOCIADO A MANTDATOS (FORMULARIO DE LA BASE DE DATOS)

Para la implementación del proyecto se empleó un archivo de acceso aleatorio conocido también como archivo de acceso directo que consiste en un conjunto de registros de la misma longitud, cada uno de ellos identificado por un único número lo que permite un acceso directo al registro.

Los pasos a seguir en la manipulación de archivos aleatorios son:

- Definición de la longitud de registro.
- Apertura del archivo.
- Proceso de los datos
- Cierre del archivo.

Para crear un archivo aleatorio seguimos estos pasos:

1. Definimos los campos del registro, indicando el tipo de dato y longitud.
2. Declaramos las variables tipo registro.
3. Abrimos el archivo en modo de acceso aleatorio y especificamos la longitud del registro. (Sentencia Open).

4. Leemos los datos del registro y grabamos el contenido en el archivo (Sentencia Put).

Para declarar la estructura de los registros que tendrá el archivo **DatPacientes.dat** es necesario crear un módulo. Para ello, seleccionamos el comando **Agregar** módulo del menú **Proyecto**. En este módulo digitamos la estructura del archivo DatPacientes.dat.

```
Type registro
    Código As String * 10
    Nombre As String * 50
    Dirección As String * 50
    Teléfono As String * 50
    Cuerpo As String * 50
    AVI As String * 10
    JPG(6) As String * 10
    Diagnóstico As String * 200
    SW As String * 1
End Type
Global DatPacientes As registro
Global TotalRegistros As Integer
```

Para tener el control del texto que presentamos en la ficha médica formamos a partir del texto cadenas de caracteres que no supere el ancho deseado y al mismo tiempo ir escribiendo una cadena a continuación de otra.

Para realizar el proceso descrito, vamos a escribir un nuevo procedimiento denominado FormatoTexto. Este proceso será invocado desde el procedimiento **txtCampo.**

Ejecutamos la orden **Módulo** del menú **Insertar** para añadir a la aplicación un nuevo módulo FormatoTexto, el parámetro Texto se refiere al área que aún nos falta por escribir; este parámetro es pasado por valor para que no se modifique el origen.

Denominamos Línea a la cadena de caracteres mayor que no supere el ancho deseado (en nuestro proyecto, la cadena es de palabras completas) y que, por lo tanto, dará lugar a la siguiente línea de caja. La función UnaPalabra devuelve la siguiente palabra de Texto, que añadiremos a Línea siempre y cuando la longitud actual de Línea más de la palabra no supere el ancho; si esto sucede, entonces escribimos Línea y la palabra devuelta por la función UnaPalabra pasará a ser la primera de la siguiente Línea. Para saber si Linéa más la siguiente palabra supera el ancho de la caja, utilizamos la variable temporal

denominada Temp. Esta variable almacena Línea más la siguiente palabra que pretendemos añadir.

Con todo lo dicho, el procedimiento FormatoTexto queda como se indica a continuación:

```
Public Sub FormatoTexto(ByVal Texto As String)
  Dim línea As String, SiguientePalabra As String
  Dim temp As String
  SiguientePalabra = UnaPalabra(Texto)
  Do
    temp = línea & SiguientePalabra
    If (Len(temp) > 70) Then
      Printer.Print Tab(35); línea
      línea = SiguientePalabra
    Else
      línea = temp
    End If
    SiguientePalabra = UnaPalabra(Texto)
  Loop Until (SiguientePalabra = "")
  Printer.Print Tab(35); línea
End Sub
```

La función UnaPalabra devuelve la primera palabra de la cadena Texto al mismo tiempo que la elimina de dicha cadena. De esta forma, cada vez que la función UnaPalabra es invocada, devuelve la siguiente palabra. Para tomar la siguiente palabra de Texto, lo único que tenemos que hacer es buscar donde se encuentra el siguiente espacio en blanco. De esta tarea se encarga la función **Instr.(X$,Y$)**, que devuelve la posición que ocupa el primer carácter de Y$ en X$. En nuestro caso, la cadena **Y$** es un espacio blanco, y X$ es el texto que aún queda por escribir. Si Y$ no se encuentra en X$, la función **InStr** devuelve un 0.

```
Public Function UnaPalabra(Texto As String) As String
  If (InStr(Texto, " ") = 0) Then
    UnaPalabra = Texto
    Texto = ""
  Else
    UnaPalabra = Left(Texto, InStr(Texto, " "))
    Texto = Right(Texto, Len(Texto) - InStr(Texto, " "))
  End If
End Function
```

Definimos primero en el formulario dos variable enteras que son utilizadas como contadores.

```
Dim SW As Integer
Dim j As Integer
```

El procedimiento asociado al formulario lo sitúa en el centro de la pantalla y abre el archivo aleatorio utilizando la sentencia **Open**.

```
Private Sub Form_Load()
  'Sitúa el formulario en el centro de la pantalla
  Move (Screen.Width - Width) / 2, (Screen.Height - Height) / 2
  'Abre el archivo DatPacientes.dat
  Open "C:\Archivos de programa\Microsoft Visual Studio\VB98\
DatPacientes.dat" For Random As #1 Len = Len(DatPacientes)
  'Llama al procedimiento LLenarLista
  LLenarLista
End Sub
```

El procedimiento **LlenarLista** debe llenar el control ListBox con los códigos de los registros.

```
Public Sub LLenarLista()
  numreg = LOF(1) \ Len(DatPacientes)
  'Las siguientes instrucciones llenan la lista con los códigos de los registros
  If numreg <> 0 Then
    lstCódigo.Clear
    For i = 1 To LOF(1) \ Len(DatPacientes)
      Get #1, i, DatPacientes
      If DatPacientes.SW = "1" Then lstCódigo.AddItem DatPacientes.Código
    Next
  Else
    Exit Sub
  End If
End Sub
```

Cuando hacemos clic en el botón **Imprimir** realizamos una serie de instrucciones para colocar el texto en la hoja de la ficha médica. Presentamos el nombre del paciente, dirección , parte del cuerpo, diagnóstico y la impresión de la imagen seleccionada.

```
Private Sub impresión_Click()
  Printer.Print
  Printer.Print
  Printer.Print
```

```
  Printer.FontSize = 12
  Printer.Print Tab(13); "Nombre Paciente"; Tab(33); ": " &
DatPacientes.Nombre
  Printer.Print Tab(13); "Dirección"; Tab(33); ": " & DatPacientes.Dirección
  Printer.Print Tab(13); "Parte Cuerpo"; Tab(33); ": " & DatPacientes.Cuerpo
  Printer.Print
  Printer.Print Tab(13); "Diagnóstico"; Tab(33); ": "
  Call FormatoTexto(DatPacientes.Diagnóstico)
  Printer.PaintPicture Picture1.Picture, Printer.Width / 4 + Printer.Width / 40,
Printer.Height / 4 + Printer.Height / 60
  Printer.EndDoc
End Sub
```

Asociamos un procedimiento al control **ListBox** accionando por el suceso **Click**. Cada vez que el usuario haga clic sobre uno de los códigos que muestra ese control, los datos relacionados con el código deben aparecer en los cuadros de texto

Este procedimiento se ejecutará cada vez que seleccione un código en la lista

```
Private Sub lstCódigo_Click()
  For i = 1 To LOF(1) \ Len(DatPacientes)
```

```
    Get #1, i, DatPacientes
    If DatPacientes.Código = lstCódigo.Text Then
      txtCampo(0).Text = DatPacientes.Código
      txtCampo(1).Text = DatPacientes.Nombre
      txtCampo(2).Text = DatPacientes.Dirección
      txtCampo(3).Text = DatPacientes.Teléfono
      txtCampo(4).Text = DatPacientes.Cuerpo
      txtCampo(5).Text = DatPacientes.AVI
      For j = 0 To 6
        txtCampo(6 + j).Text = DatPacientes.JPG(j)
      Next
      txtCampo(13).Text = DatPacientes.Diagnóstico
    End If
  Next
  cmdGuardar.Enabled = False
  cmdModificar.Enabled = False
  cmdEliminar.Enabled = True
End Sub
```

Asociamos un procedimiento al botón de comando **Ingresar** accionado por el suceso **Click**. El procedimiento debe habilitar e inhabilitar los botones Guardar,

Modificar y Eliminar; además, debe llamar al procedimiento LimpiarCuadrosDeTexto.

```
Private Sub cmdIngresar_Click()
  LimpiarCuadrosDeTexto
  cmdGuardar.Enabled = True
  cmdModificar.Enabled = False
  cmdEliminar.Enabled = False
End Sub
```

Asociamos un procedimiento al botón de comando **Guardar** accionado por el suceso **Click**. El procedimiento debe guardar los nuevos datos en el archivo DatPacientes.dat.

```
Private Sub cmdGuardar_Click()
  'Ingresa nuevos registros en el archivo DatPacientes.dat
  DatPacientes.Código = txtCampo(0).Text
  DatPacientes.Nombre = txtCampo(1).Text
  DatPacientes.Dirección = txtCampo(2).Text
  DatPacientes.Teléfono = txtCampo(3).Text
  DatPacientes.Cuerpo = txtCampo(4).Text
  DatPacientes.AVI = txtCampo(5).Text
```

```
  For j = 0 To 6
    DatPacientes.JPG(j) = txtCampo(6 + j).Text
  Next
  DatPacientes.Diagnóstico = txtCampo(13).Text
  DatPacientes.SW = 1
  TotalRegistros = LOF(1) \ Len(DatPacientes)
  Put #1, TotalRegistros + 1, DatPacientes
  LLenarLista
  LimpiarCuadrosDeTexto
  cmdGuardar.Enabled = False
End Sub
```

Asociamos un procedimiento al botón de comando **Modificar** accionado por el suceso **Click**. El procedimiento debe guardar los cambios realizados en los campos de un registro.

```
Private Sub cmdModificar_Click()
  Dim i As Integer
  Dim encontrado As Boolean
  For i = 1 To LOF(1) \ Len(DatPacientes)
    Get #1, i, DatPacientes
    If DatPacientes.Código = lstCódigo.Text Then
```

```
      encontrado = True
      posición = i
      Exit For
    End If
  Next i
  If encontrado Then
    DatPacientes.Código = txtCampo(0).Text
    DatPacientes.Nombre = txtCampo(1).Text
    DatPacientes.Dirección = txtCampo(2).Text
    DatPacientes.Teléfono = txtCampo(3).Text
    DatPacientes.Cuerpo = txtCampo(4).Text
    DatPacientes.AVI = txtCampo(5).Text
    For j = 0 To 6
      DatPacientes.JPG(j) = txtCampo(6 + j).Text
    Next
    DatPacientes.Diagnóstico = txtCampo(13).Text
    Put #1, posición, DatPacientes
  End If
  LLenarLista
  cmdModificar.Enabled = False
End Sub
```

Asociamos un procedimiento al botón de comando **Eliminar** accionado por el suceso **Click**. Cuando el usuario selecciones un código en el control ListBox y haga clic en el botón Eliminar, los datos relacionados con el código deben quedar eliminados del archivos DatPacientes.dat.

```
Private Sub cmdEliminar_Click()
  Dim i, posición As Integer
  Dim encontrado As Boolean
  encontrado = False
  For i = 1 To LOF(1) \ Len(DatPacientes)
    Get #1, i, DatPacientes
    If DatPacientes.Código = lstCódigo.Text Then
      encontrado = True
      posición = i
      Exit For
    End If
  Next i
  If encontrado Then
    Get #1, posición, DatPacientes
    DatPacientes.SW = 0
    Put #1, posición, DatPacientes
  End If
```

```
  LLenarLista
  LimpiarCuadrosDeTexto
  cmdEliminar.Enabled = False
  cmdModificar.Enabled = False
End Sub
```

Asociamos un procedimiento al array de controles TextBox accionado por el suceso **Change**. Cuando el usuario realice cualquier cambio en uno de los controles que forman parte de array, este procedimiento se ejecutará y habilitará los botones de comando Modificar y Eliminar.

```
Private Sub txtCampo_Change(Index As Integer)
  cmdModificar.Enabled = Not cmdGuardar.Enabled
  cmdEliminar.Enabled = False
End Sub
```

Asociamos un procedimiento al array de controles TextBox accionado por el suceso **Click**. Cuando el usuario coloque el puntero sobre el nombre de video o sobre algún nombre de imagen, este procedimiento se ejecutará para validar los nombres seleccionados, cuando el archivo seleccionado se encuentra se habilitara los MCI y la opción de Imprimir, caso contrario limpia el cuadro de imagen y desactiva los controles.

```
Private Sub txtCampo_Click(Index As Integer)
  Dim Nombre As String
  If Index < 5 Or Index > 12 Then
    impresión.Enabled = False
    Picture1.Picture = LoadPicture()
  End If
  If Index = 5 Then
    Nombre = "D:\imágenes ecógrafo\" + RTrim(DatPacientes.AVI) + ".avi"
    On Error GoTo ningún_avi
    MMControl1.FileName = Nombre
    MMControl1.Command = "Open"
    GoTo avi_encontrado
ningún_avi:
      Picture1.Picture = LoadPicture()
avi_encontrado:
  End If
  If Index > 5 And Index < 13 Then
    If RTrim(DatPacientes.JPG(Index - 6)) <> "" Then
      Nombre = "D:\imágenes ecógrafo\" + RTrim(DatPacientes.JPG(Index - 6))
+ ".jpg"
      On Error GoTo ningún_archivo
      MMControl1.Command = "close"
      Picture1.Picture = LoadPicture(Nombre)
```

```
      impresión.Enabled = True
      GoTo archivo_encontrado
ningún_archivo:
        Picture1.Picture = LoadPicture()
        impresión.Enabled = False
archivo_encontrado:
      Else
        Picture1.Picture = LoadPicture()
        impresión.Enabled = False
    End If
  End If
End Sub
```

Asociamos otro procedimiento al array de controles TextBox accionado por el suceso **KeyPress**. Cuando el usuario Introduzca datos en uno de estos controles y presione la tecla Enter, el enfoque debe pasar al siguiente control TextBox.

```
Private Sub txtCampo_KeyPress(Index As Integer, KeyAscii As Integer)
  If KeyAscii = 13 And Index < 13 Then txtCampo(Index + 1).SetFocus
End Sub
```

Agregamos el procedimiento LimpiarCuadrosDeTexto al programa. Este procedimiento debe limpiar todos los cuadros de texto.

```
Sub LimpiarCuadrosDeTexto()
  'Limpia los cuadros de texto del formulario
  For i = 0 To 13
    txtCampo(i).Text = ""
  Next
  txtCampo(0).SetFocus
End Sub
```

Asociamos un procedimiento al botón de comando **Salir** accionado por el suceso **Click.**

```
Private Sub cmdSalir_Click()
  Unload Me
End Sub
```

Aplicamos la acción **Unload** al formulario donde cerramos el archivo DatPacientes.dat.

```
Private Sub Form_unLoad(Cancel As Integer)
  'Cierra el archivo DatPacientes.dat
  Close
End Sub
```

CAPITULO TRES

EJECUCIÓN DEL PROGRAMA

3.1. CREACIÓN DE ARCHIVOS EJECUTABLES

Una vez que la aplicación tiene el aspecto deseado y que su ejecución se realiza correctamente, se puede crear un archivo ejecutable que permita que dicha aplicación se ejecute fuera del entorno de Visual Basic. Para ello, en el menú Archivo, hacemos clic en la opción Generar **eco_pc.exe** Visual Basic muestra el cuadro de dialogo Generar proyecto.

Luego de seleccionar una carpeta y de introducir un nombre, hacemos clic en el botón Aceptar. Visual Basic creará un programa ejecutable y lo almacenará en la carpeta especificada.

Para ejecutar este archivo desde Windows, lo localizamos en el Explorador de Windows y hacemos clic sobre él. También podremos crear un acceso directo para este archivo en la ventana del escritorio. Para ello, en el explorador de Windows, situamos el puntero del mouse sobre el archivo ejecutable, presionamos el botón derecho del mouse y manténgalo así mientras lo arrastra hacia el escritorio de Windows. Soltamos el botón del mouse y Windows le mostrará un menú contextual. En este menú seleccione el comando **Crear iconos de acceso directo aquí.** Windows introducirá un icono en el escritorio sobre el que podrá hacer doble clic para ejecutar el archivo ejecutable.

3.2. CREACION DE PELICULAS DIGITALES CON LA TARJETA CAPTURADORA DE VIDEO.

El video es digitalizado por la capturadora de video digital la cual es capturada por medio de una secuencia de video (o video clip) usando el programa de la capturadora de video. Este programa esta suministrado en el CD ROM de instalación de la tarjeta capturadora de video y usa video de Microsoft para compatibilidad de Windows la cual esta construido dentro de Windows 95.

La capturadora de video crea un archivo de video con extensión AVI en el disco duro. Este archivo contiene video digitalizado y, opcional audio.

Esta capturadora de video da alta calidad de captura para almacenarlo como video digital descomprimido en el archivo AVI. Esto se compara con los métodos de compresión tales como JPEG y MPEG, donde la perdida de calidad de video es aceptada para así reducir la cantidad de datos almacenados. La captura de video no comprimida crea alta calidad de video digital pero requiere un diseño de sistema optimo.

CAPITULO CUATRO

SISTEMA EN FUNCIONAMIENTO

4.1. DIAGRAMA DE BLOQUES GENERAL

Figura 4.1. Diagrama de bloques del sistema

4.2. DIAGRAMA DE FLUJO

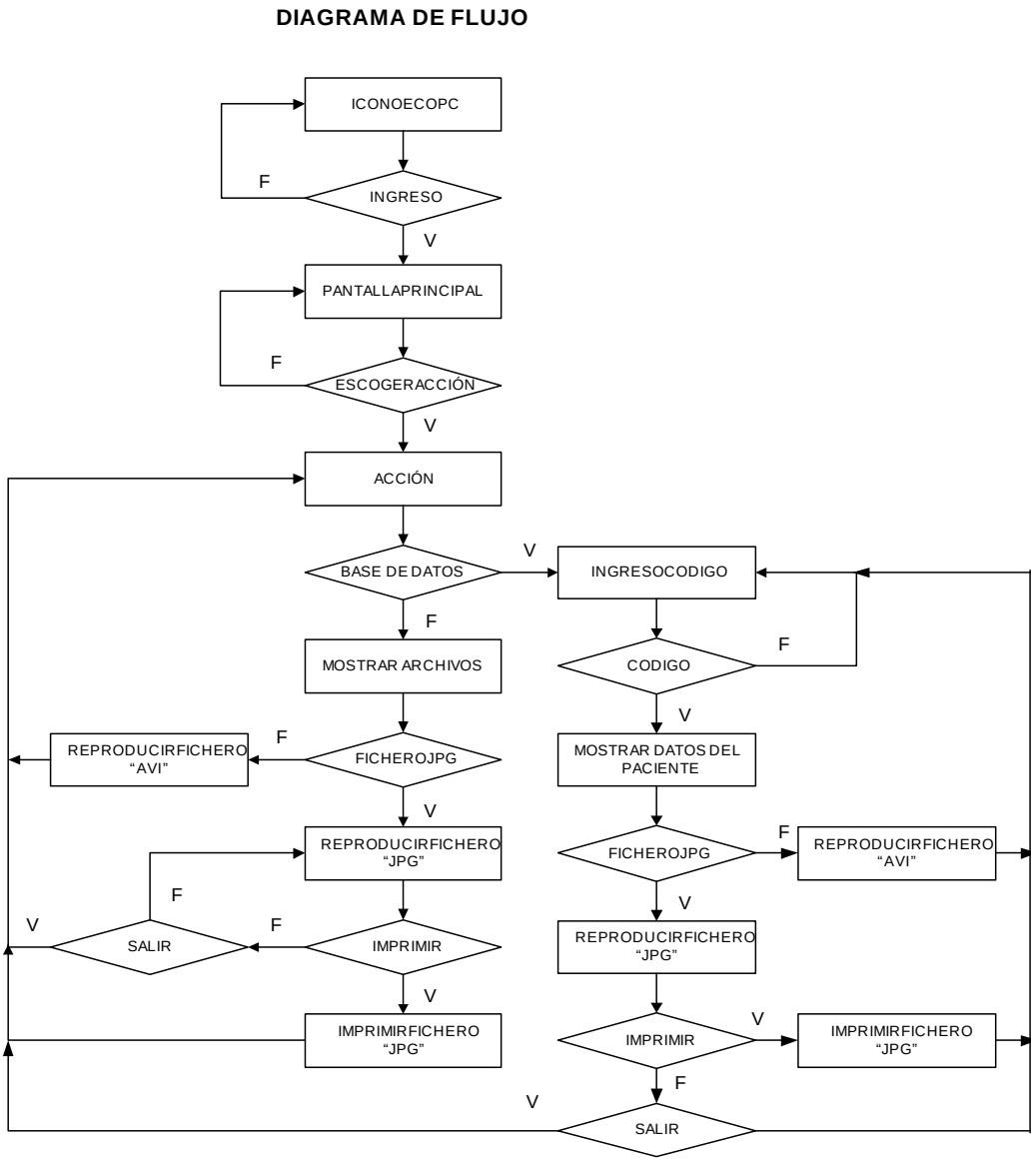

Figura 4.2. Diagrama de flujo del sistema

4.3. PRUEBAS REALIZADAS

Con el sistema descrito anteriormente se realizo una simulación de operación usando un dispositivo de audio y video con salida de video compuesto entrelazado, tal es el caso de un VHS que se asemeja la salida de video de un ecógrafo.

Figura 4.3. Diagnóstico por medio de un ecosonógrafo

En principio la imagen y el video que proporcionaba la capturadora de video eran de calidad regular, por lo que manipulando el software de instalación que viene con la tarjeta capturadora esta imagen en formato JPG y el video en formato AVI resultó con una mejor resolución.

Los tamaños de imagen capturados pueden ser de 160x120 o 320x240 píxeles.

Figura 4.4. Imagen capturada de 320x240 píxeles

Para que sea mejor la compatibilidad entre tamaño y tramos por segundos capturados usamos una imagen de 320x240. Con un procesador 486/66 Mhz y un disco duro con una velocidad razonable rápida, sería capaz de capturar 30 tramas por segundo usando el formato BTYUV. Este formato usa la velocidad de compresión de 25:1 para “llover” 24 bits por píxel de video.

El número de buffer de video dependerá bajo cuanta memoria RAM tiene el computador. El máximo número es de 1000.

El consumo de espacio de disco duro depende del tamaño del video, el formato de la imagen más el número de tramas por segundo. Usando el formato BTYUV, la siguiente tabla muestra el espacio consumido del disco duro.

Tabla IV. Capacidad consumido del video en el disco duro

Tamaño de imagen	Tramas/segundos	Bytes por segundo	1 minuto de video
160x120	15	425K	26M
320x240	15	850K	52M
320x240	30	1.7M	104M

La ficha médica que resulta después de revisar la base de datos del paciente y seleccionar una de la imágenes del paciente es la siguiente.

Nombre Paciente : Mercedes Posligua Coello
Dirección : ciudadela Coviem manzana 40 villa 11
Parte Cuerpo : corazón

Diagnóstico :

Espacio destinado para el diagnóstico médico detallado del especialista en la sección de ecografía.

4.4. VENTAJAS

La principal ventaja que tiene este sistema es la no-utilización de un papel especifico para ecografía, papel térmico. El sistema de la computadora, la capturadora de video, la interfase gráfica de usuario, el ecógrafo e impresora utiliza papel común para la impresión, lo que resulta económico en el momento de hacer un informe médico para uso de un diagnóstico.

Como el programa posee una base de datos, se automatizó el proceso de control y seguimiento del uso de carpetas por este sistema actual usando la tecnología del futuro.

Si se tiene una pequeña red de datos para computadoras este sistema puede ser implementado mediante un servidor de archivos de base de datos para que puedan ser ejecutados en distintos puntos remotos.

4.5. COSTOS vs BENEFICIOS

Los costos de este trabajo se reflejan en los requerimientos de hardware, la tarjeta capturadora de video, licencias y tiempo de implementación del trabajo.

Por tal motivo ponemos a disposición la siguiente tabla que indica los costos a la fecha actual para la realización del trabajo:

Tabla V. Costos y beneficios

Materiales	**Costos**	**Beneficios**
Un computador con requerimientos mínimos de hardware	400.00	Necesario para cualquier utilidad
Una tarjeta capturadora de video	50.00	Edición, compresión de video
Una licencia de instalación Visual Basic	45.00	Diseños de futuros programas
Desarrollo del programa	100.00	Todas las ventajas mencionadas

Con un costo aproximado de $ 595.00 incluyendo la base de datos, el proyecto puede ser de uso comercial en pequeñas dispensarios o consultorios médicos, para su fácil control y seguimiento de los pacientes.

CONCLUSIONES Y RECOMENDACIONES

Como parte final del desarrollo de este proyecto a continuación se plantean las conclusiones basadas en las experiencias de las pruebas realizadas.

- Con las nuevas tendencias tecnológicas se realiza una aplicación de interacción hombre-computadora creando una interfase gráfica de usuario que permita escoger y almacenar la imagen o vídeo mejor descriptiva por medio del enlace que conecte la salida de audio y video entrelazado de un ecosonógrafo con la entrada de la tarjeta de audio y video entrelazado acoplada a una computadora personal.

- Dado la cambiante tecnología se buscó alternativas de aplicación en el campo médico surgió la conjunción de la medicina, electrónica e informática para dar como resultado la telemedicina.

- Realizar, por medio de la computadora una base de datos para llevar un control de la información obtenida de un paciente usando la misma interfase gráfica de usuario.

- Reducción de los costos en la impresión de una sesión de ecografía, a una simple hoja de papel pero de alta resolución de imagen por la computadora.

ANEXOS

ANEXO I

DEFINICIÓN DE CONCEPTOS PARA EL DESARROLLO DEL PROGRAMA EN EL LENGUAJE DE PROGRAMACIÓN MICROSOFT VISUAL BASIC

Una aplicación de Visual Basic es conducida por sucesos y orientada a objetos, es decir, por clic del ratón, un mensaje DDE o un tiempo transcurrido.

En Visual Basic, un objeto tiene asociados datos y se denominan propiedades, también tiene asociados métodos que son los procedimientos.

Podemos categorizar los controles, esto es:

1. Los controles estándar, esto es las cajas de texto, botones, etc.
2. Los controles personalizados, esto es los ocx

A continuación damos algunos conceptos básicos del entorno de programación en Visual Basic.

Procedimiento: es el código que se une a un objeto, o también llamado procedimiento conducido por un suceso.

Formulario: Un formulario es una ventana en la que se “dibujan” los controles y que permite a los usuarios llevar a cabo las funciones asociadas a la aplicación

Controles: Los controles son objetos que es “dibujan” sobre un formulario, tales como etiquetas, cuadros de texto, botones de comando, marcos, listas, temporizadores, etc.

Botón de comando: CommandButton, crea un botón en el que el usuario puede hacer clic para ejecutar un comando.

Menú: Desde el cual el usuario puede ejecutar de manera fácil un comando.

Caja de imagen: PictureBox, se utiliza para visualizar imágenes gráficas.

Control MCI: Proporciona una forma fácil de grabar y reproducir ficheros multimedia. Está formado por un conjunto de botones que permiten enviar órdenes MCI a dispositivos tales como tarjetas de sonido, secuenciadores MIDI, unidades de CD-ROM, reproductores CD de audio, reproductores videodiscos y grabadoras y reproductoras de cintas de video.

Cuadro de lista de unidades: DriveListBox, se utiliza para visualizar una lista de las unidades de disco disponibles para que el usuario pueda seleccionar una.

Cuadro de lista de directorios: DirListBox, se utiliza para visualizar una lista de carpetas de un dispositivo seleccionado en los que el usuario puede moverse.

Cuadro de lista de archivos: FileListBox, se utiliza para visualizar una lista de archivos contenidos en la carpeta seleccionada a los que el usuario puede acceder.

Marco: Frame, se utiliza para agrupar objetos relacionados entre sí. Para agrupar controles, dibuje primero el marco y luego dibuje los controles dentro del marco.

Cuadro de dialogo común: CommonDialog, permite realizar operaciones comunes como abrir y guardar archivos, establecer opciones de impresión y seleccionar colores y fuentes.

1.1 Aplicación Visual Basic

Existen varias aplicaciones que se generan cuando realizamos un proyecto en el lenguaje de programación de Visual Basic, cada uno con su respectiva extensión.

Formularios	(.frm)
Módulos	(.bas)
Clases	(.cls)
Recursos	(.res)
Controles ocx	(.ocx)
Controles vbx	(.vbx)

Cuando se guarda la aplicación tiene extensión .vbp

ANEXO II

PATRONES DE VIDEO

El ojo humano es capaz de percibir imágenes mediante receptores ubicados en la retina. Hay dos tipos de receptores, según su función: los bastones, encargados de percibir imágenes en blanco y negro, y los conos, a cargo de la percepción del color. Si nos concentramos en el estudio de los conos, veremos que hay de tres tipos: los que reaccionan frente a la luz roja, los que lo hacen frente a luz verde y finalmente los que son sensibles a la luz azul. Solo se perciben tres colores, sin embargo nosotros "vemos" todos los colores que nos rodean.

Aquí se hace evidente una regla básica del color: para conocer la información de color de un objeto basta con tener la proporción de los tres colores básicos: Rojo, Verde y Azul (RGB). Por esta razón a dichos colores se los conoce como Colores Primarios, ya que con la combinación de los mismos se pueden obtener todos los demás.

Cabe hacer una aclaración, ya que tal vez alguna vez se nos haya dicho que los primarios son el rojo, amarillo y azul. Es más, si observamos los colores utilizados por las impresoras de tinta, muy comunes en la actualidad, veremos que utilizan tres

cartuchos, magenta, amarillo y cian. No utilizan los primarios, y sin embargo imprimen “a todo color”.

La “confusión” resulta de la existencia de dos grupos de colores primarios: los Primarios Aditivos y los Primarios Sustractivos. Para entender la diferencia entre ambos veamos un par de ejemplos.

Supóngase que iluminamos una pared blanca con una luz verde. Obviamente se verá verde, ya que este es el color que refleja la pared y llega a nuestros ojos. Si ahora cambiamos la luz a rojo, igualmente veremos el color rojo. Pero si iluminamos al mismo tiempo con verde y rojo, el color que percibiremos será el amarillo, el cual resulta de la suma o adición de los colores verde y rojo. Como se observa, la suma de estos colores resulta en la generación de un color secundario, por esto se denominan primarios aditivos.

Ahora supongamos que pintamos un papel blanco con pintura amarilla, y lo iluminamos con luz blanca (contiene todos los colores). Obviamente se ve amarillo. ¿Por qué?. Porque la pintura retiene (sustrae) todos los componentes de la luz blanca que la ilumina y devuelve solo el amarillo, el cual llega a nuestros ojos. Si ahora agregamos pintura cian (celeste intenso), el color resultante es verde. Quiere decir que la mezcla ha retenido todos los colores de la luz blanca menos el verde. ¿Cómo se explica esto?. Veamos algunas ecuaciones.

Si sumamos todos los colores primarios aditivos obtenemos blanco:

LUZ ROJA + LUZ VERDE + LUZ AZUL = LUZ BLANCA

Si los sumamos de a dos:

LUZ ROJA + LUZ VERDE = LUZ AMARILLA

LUZ ROJA + LUZ AZUL = LUZ MAGENTA

LUZ VERDE + LUZ AZUL = LUZ CIAN

De aquí surge una relación clave en nuestro análisis: los llamados “colores primarios sustractivos” resultan de combinar dos a dos los primarios aditivos.

Entonces, cuando vemos la pintura amarilla en realidad estamos viendo luz roja y luz verde combinadas. Quiere decir que la pintura amarilla sustrae el azul de la luz blanca y devuelve los otros dos primarios.

Por otra parte, la pintura cian retiene el rojo y devuelve luz verde y luz azul.

Entonces, ahora es evidente el resultado de nuestro experimento. Si mezclamos pintura amarilla (retiene el azul) con pintura cian (retiene el rojo), el único color que puede salir de esta mezcla es el verde, el cual es justamente el color que vemos.

¿Qué ocurre si mezclamos los tres colores primarios sustractivos?. Obviamente se ve negro, ya que todas las “luces” son retenidas:

CIAN + MAGENTA + AMARILLO = NEGRO

Queda aún una pregunta sin resolver. ¿Por qué habitualmente se dice que los colores primarios son rojo, amarillo y azul?. El motivo es sencillo. Desde niños se nos enseña el color mediante la pintura, y en este campo ya sabemos que trabajamos con los primarios sustractivos, amarillo, magenta y cian. A estos últimos, por ser rojizo y azulado, respectivamente, se los llama “rojo” y “azul”.

2.1. CARACTERÍSTICAS DE LAS SEÑALES DE VIDEO

Son formas de onda eléctricas que permiten transmitir imágenes en movimiento de un lugar a otro.

El Tubo de rayos catódicos (TRC) no es un dispositivo lineal, sino que produce luz de acuerdo con una ley de potencia con un índice de aproximadamente 2,2, esta característica no lineal se conoce como gamma. En televisión la exploración consiste en barridos horizontales rápidos combinados con barridos verticales más lentos, de manera que la imagen quede explorada en líneas. En el sistema entrelazado, la velocidad de barrido vertical se duplica de manera que

haya espacios entre las líneas escaneadas. El primer campo comience con una línea completa y acabe a la mitad de otra línea y el segundo campo comience a mitad de una línea y finalice con una línea completa. Las líneas de ambos campos se entrelazan verticalmente de forma automática.

Los tipos principales de video analógico. Las señales rojas, verdes y azules salen de los censores de la cámara, siendo necesario un ancho de banda completo. Si se obtiene una señal de luminancia mediante una suma equilibrada de R, G, B, necesitará el ancho de banda total, pero las señales de diferencia de color R-Y y B-Y en un sistema de modulación de subportadora permite la transmisión de color en el mismo ancho de banda que el monocromo.

Dado que las señales rojas, verdes y azules representan directamente parte de la imagen, este sistema se conoce como video compuesto.

El ojo humano se basa en el brillo para obtener detalles, y se necesita mucha menos resolución en la información del color. El R, G, y B constituyen una matriz para conformar la señal Y de luminancia que tiene un ancho de banda completo. Las señales de diferencias de color no necesitan el mismo ancho de banda que Y, ya que la agudeza del ojo no llega a la visión del color. La mitad o un cuarto del ancho de banda será suficiente dependiendo de la aplicación.

2.2. LA IMAGEN A COLOR EN TELEVISIÓN

La imagen en un televisor a colores se forma mediante la emisión de luz resultante de la excitación de la película de fósforo, que recubre internamente la pantalla, al ser alcanzada por un haz de electrones que barre periódicamente la superficie visible. Si hablamos de “emisión de luz”, inmediatamente debemos pensar en procesos “aditivos”, lo cual nos lleva a concluir que en televisión los colores primarios son el Rojo, Verde y Azul (RVA o en inglés RGB).

Efectivamente, dentro del tubo de televisión se emiten tres haces de electrones, destinados cada uno a excitar una franja de fósforo en la pantalla, la cual responderá emitiendo un color característico al fósforo empleado. Naturalmente, como no podía ser de otro modo, estos colores son Rojo, Verde y Azul.

Todos los demás colores (y digo realmente “todos”) se pueden obtener combinado estos tres primarios en distintas proporciones.

2.3. GENERADOR DE BARRAS BÁSICO

Un generador de barras de color básico podría hacer lo siguiente:

- Tener tres salidas, una para cada color primario
- Cada una de estas salidas se conecta a la correspondiente entrada del televisor

- Si se tiene un equipo que genere combinaciones de sus salidas, según la siguiente tabla:

Azul	1	0	1	0	1	0	1	0
Rojo	1	1	0	0	1	1	0	0
Verde	1	1	1	1	0	0	0	0

Figura B2.1. Combinación de colores primarios para formar barras de colores básicos.

En esta tabla un “1” significa “presencia” del color, en tanto que un “0” es su “ausencia”. En la práctica, estos “unos” y “ceros” se representan por niveles de tensión, por ejemplo 5V y 0V respectivamente.

La mayoría de los equipos de televisión y video no poseen entradas directas de Rojo, Verde y Azul, estas quedan reservadas para monitores destinados al campo profesional. Lo habitual es que los equipos hogareños tengan una entrada de “Video Compuesto”, denominada usualmente como “VIDEO IN”.

2.4. RUIDO EN LAS IMÁGENES

Los parámetros que determinan la calidad de la visualización de un documento sobre la pantalla de un monitor son: Luminancia (agudeza, diferenciación), Rango dinámico, Ruido (temporal, espacial), Distorsión (astigmatismo, distorsión de campo) y Resolución (espacial, contraste).

En el proceso de generación de imágenes, algunos ruidos son introducidos en las imágenes. Generalmente, los píxeles con ruido aparecen como puntos con niveles de gris bien diferentes de su vecindad (oscuros -negros- o saturados -blancos-). Estos puntos pueden aparecer distribuidos aleatoriamente o en forma sistemática (fajas verticales y horizontales). Las causas de dicho ruido pueden ser fallas de los detectores, limitaciones del sistema electrónico del censor, entre otros.

A la pregunta: ¿puede el radiólogo especialista diagnosticar sobre la pantalla de un monitor?, la respuesta es obvia: depende de la calidad de la imagen que visualiza.

En el concepto de "calidad" es necesario incluir todo aquello que el radiólogo necesita [confianza y seguridad en el sistema] para sentirse en condiciones de elaborar el informe. De otra forma, no solamente placas "complicadas" como fracturas sin desplazamiento, pneumotorax, enfermedades intersticiales del

pulmón, etc., sino muchas otras con menor dificultad, no podrán ser correctamente informadas; y en muchos casos, ante las razonables dudas, el especialista no realizará el informe.

El objetivo de eliminar o reducir los puntos de ruido en la imagen. Es definir dos umbrales: Umbral Inferior y Umbral Superior.

Se deberá definir los valores de los umbrales que serán utilizados en la caracterización de los puntos con ruido.

Para saber si un punto en la imagen P(i, j) (i es la línea y j es la columna) es ruido o no, se analizan solamente sus vecinos superior P(i-1, j) e inferior P(i+1, j).

2.5. Umbral Inferior

Un punto será considerado ruido cuando su nivel de gris sea menor de los niveles de gris de sus dos puntos vecinos de las líneas de arriba y de abajo por una diferencia mayor que el umbral inferior definido. En caso de que el punto corresponda a ruido, será substituido por la media de los dos puntos vecinos. El valor "default" para este parámetro es 8.

2.6. Umbral Superior

Un punto será considerado ruido en caso de que su nivel de gris sea superior al de los dos puntos vecinos de arriba y de abajo (línea de arriba y de abajo) por una diferencia mayor que el umbral superior. En este caso el punto también será substituido por la media de los dos puntos vecinos. El valor "default" para este parámetro es 25.

BIBLIOGRAFÍA

- Cevallos Sierra Fco. Javier, **Enciclopedia de Microsoft Visual Basic 4.0**, México 1997, pp 1-954, Editorial Omega, Biblioteca Protcom ESPOL - Peñas.

- Luna Rubén, **Programación en Visual Basic 6.0**, 1ra Edición, Lima- Perú, 2000, pp. 1-420, Editorial Macro E.I.R.L, Librería Cervantes.

- Khader Michael, Barnes Williams, **Telecommunications Systems and Technology,** Cap 12, New Jersey, Prentice Hall, Biblioteca Central de Ingeniería, ESPOL.