UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	IDE DEL LENGUAJE (DISEÑO DE GUI PARA CALCULADORA BASICA)	PRÁCTICA NÚMERO	1
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

IDE (entorno de desarrollo integrado)es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

2.- OBJETIVO (COMPETENCIA):

Desarrollar las habilidades para el uso eficiente del IDE del lenguaje de programacion visual seleccionado por el maestro para derrallar una interfaz de usario gráfica (GUI) de una calculadora basico, en este punto el alumno se relacionará con el lenguaje por primera ves, conocera los componentes del IDE, help, File, abrir y salvarl el projecto, crear una forma, colocará los controles necesarios para la calculadora basica, como botones y caja de texto para desplegar los numeros y los operadores de la calculadora, en la practica siguiente el maestro proporcionara el codico para los eventos y metodos de esta aplicación. (el boton de punto decimal es opcional)

3.- TEORÍA:

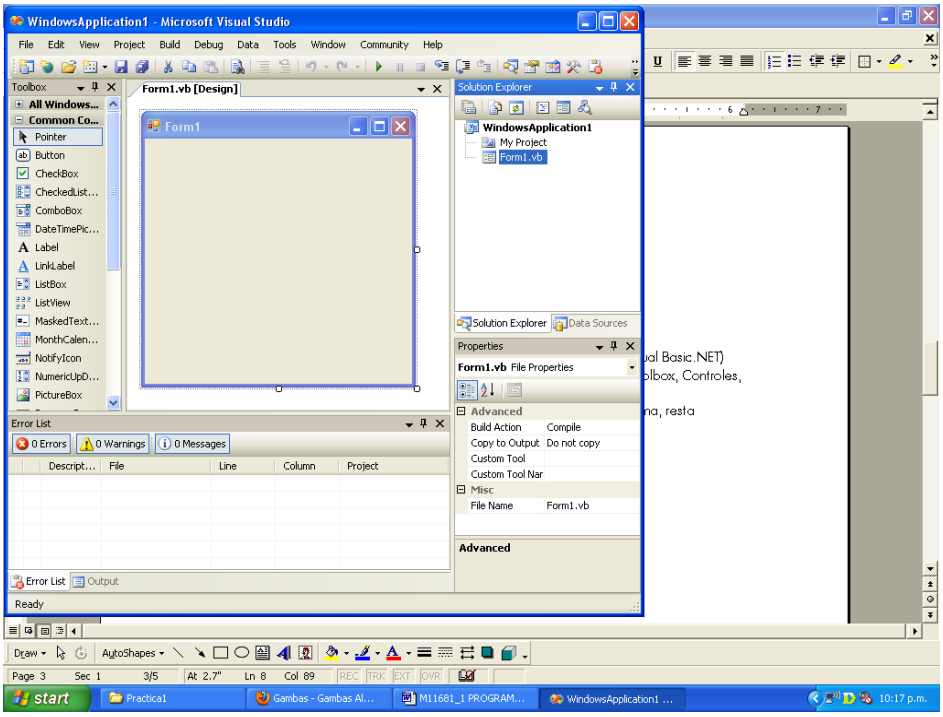
- Conocimiento básico de algoritmos.
- Introducción a IDE de Visual Basic, Java o C# en caso de Windows y en caso de linux Gambas.

Principales y posibles IDE de desarrollo para esta practica:

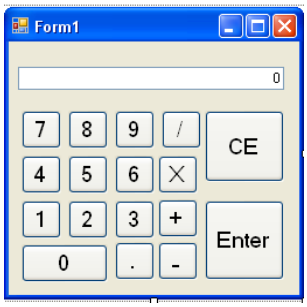
- **Visual Studio (Windows):** Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Este seria una buena opción en caso de que el laboratorio cuente con sistema operativo Windows (XP, Vista o Windows Seven).
- **Gambas (Linux)** : Gambas es un lenguaje orientado a objetos con gran número de capacidades y un entorno de desarrollo basado en un intérprete de BASIC. Se encuentra publicado bajo licencia GNU General Public Licence.. (para el desarrollo de GUI en visual basic bajo la plataforma de sistema operativo Linux (Ubuntu, debian, suse etc)

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Encendido del equipo de cómputo.
2. Acceso al sistema UABC (windows o Linux)
3. Acceso al lenguaje de programacion Visual Conveniente. (ej. Visual Basic.NET)
4. Explicación de las funciones básicas de IDE : (help, file, Forma, Toolbox, Controles, (textbox, buttons, etc.) Abrir, salvar proyecto y nombrar proyecto. Crear una forma, tomar y arraxtral los controles del toolbox. Y expliación de propiedades de cada uno de los controles mas utilizados del toolbox. (color, name, text, textalign, size, focus, tab etc).
5. 

6. Colocación de controles necesarios para Calculadora básica (suma, resta división, multiplicación, números del 0-9, Enter, clear(CE),el botón de punto decimal es opcional.



7. Una ves creado el GUI de la calculadora, el maestro proporcionará codigo ya preparado para asociar a cada uno de los eventos de click de los botones del GUI desarrollado, para mostrar como funcionaría la calculadora (este codigo es desarrollado por el maestro puesto que en esta práctica el alumno todavia no tiene todos los elementos del lenguaje).
8. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la calculadora
9. Duración de la práctica 4 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

6.- ANEXOS: Ejemplo de codigo asociado alos eventos de la calculadora basica.

(CODIGO PROPUESTA PARA VISUAL BASIC. NET:)

```
    Dim operador_listo As Boolean
    Dim calculo_nuevo As Boolean
    Dim a_str As String
    Dim a_dato As Double
    Dim b_dato As Double
    Dim num_str As String
    Dim resultado As Double
    Dim operador As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        inicializa()
    End Sub

    Private Sub unobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles unobtn.Click
        num_str = "1"
        checa_texto_actual()
    End Sub
```

```
    Private Sub dosbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles dosbtn.Click
        num_str = "2"
        checa_texto_actual()
    End Sub

    Private Sub tresbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles tresbtn.Click
        num_str = "3"
        checa_texto_actual()
    End Sub

    Private Sub cuatrobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cuatrobtn.Click

        num_str = "4"
        checa_texto_actual()
    End Sub

    Private Sub cincobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cincobtn.Click
        num_str = "5"
        checa_texto_actual()
    End Sub

    Private Sub seisbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles seisbtn.Click
        num_str = "6"
        checa_texto_actual()
    End Sub

    Private Sub sietebtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles sietebtn.Click
        num_str = "7"
        checa_texto_actual()
    End Sub

    Private Sub ochobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ochobtn.Click

        num_str = "8"
        checa_texto_actual()
    End Sub

    Private Sub nuevebtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles nuevebtn.Click
        num_str = "9"
        checa_texto_actual()
    End Sub

    Private Sub cerobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cerobtn.Click
        num_str = "0"
        If TextBox1.Text <> "0" Then
            TextBox1.Text = TextBox1.Text + "0"
        End If
    End Sub
    Private Sub checa_texto_actual()

        If operador_listo Then
            a_dato = Val(TextBox1.Text)
            TextBox1.Text = num_str
```

```
            operador_listo = False
        Else
            If calculo_nuevo Then
                TextBox1.Text = "0"
                calculo_nuevo = False
            End If
            If TextBox1.Text = "0" Then TextBox1.Text = num_str Else TextBox1.Text =
TextBox1.Text + num_str
        End If
        If TextBox1.Text = "." Then TextBox1.Text = "0."
    End Sub

    Private Sub enterbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles enterbtn.Click

        b_dato = Val(TextBox1.Text)

        Select Case operador
            Case "X"
                resultado = a_dato * b_dato
                TextBox1.Text = resultado
            Case "/"
                resultado = a_dato / b_dato
                TextBox1.Text = resultado
            Case "+"
                resultado = a_dato + b_dato
                TextBox1.Text = resultado
            Case "-"
                resultado = a_dato - b_dato
                TextBox1.Text = resultado

        End Select

        operador_listo = False
        calculo_nuevo = True

    End Sub

    Private Sub multibtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles multibtn.Click
        a_str = Val(TextBox1.Text)
        a_dato = a_str
        operador = "X"
        operador_listo = True
    End Sub

    Private Sub divibtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles divibtn.Click
        a_str = Val(TextBox1.Text)
        a_dato = a_str
        operador = "/"
        operador_listo = True
    End Sub

    Private Sub masbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles masbtn.Click
        a_str = Val(TextBox1.Text)
        a_dato = a_str
        operador = "+"
        operador_listo = True
    End Sub
```

```
    Private Sub menosbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles menosbtn.Click
        a_str = Val(TextBox1.Text)
        a_dato = a_str
        operador = "-"
        operador_listo = True
    End Sub

    Private Sub cebtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cebtn.Click
        inicializa()
    End Sub

    Private Sub inicializa()
        operador_listo = False
        a_str = ""
        a_dato = 0
        b_dato = 0
        TextBox1.Text = "0"
        calculo_nuevo = True
    End Sub

    Private Sub puntobtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles puntobtn.Click
        num_str = "."
        checa_texto_actual()
    End Sub
```

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	USO EDITOR DE CODIGO (CREAR CALCULADORA CIENTIFICA)	PRÁCTICA NÚMERO	2
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Cada lenguaje de programación visual cuenta con su Editor de codigo en su IDE, tal editor se accesa usualmente al seleccionar con doble click al Objeto o elemento o forma de nuestro diseño de GUI (nuestra interfaz grafica de usuario) donde queremos incluir codigo para un evento. En versiones mas recientes estos editores han mejorado al grado de tener funciones de auto completar el codigo que se escribe marcar errores en la sintaxis del codigo asi como mostrar los valores posibles en los campos de cada variable u objeto al momento de que se esta escribiendo el codigo.

2.- OBJETIVO (COMPETENCIA):

Desarrollar las habilidades para comensar a crear código, utilizando elementos de lenguaje ya vistos en clase, como : Declaración de variables, Operadores aritméticos, funciones básicas de trigonométría.

Para finalmente crear una calculadora Científica a partir de la práctica anterior (#1), reutilizar código y agregar nuevo para Lograr operaciones matemáticas básicas, con calculos hechos en Radianes. (la conversion a Grados es opcional)

3.- TEORÍA: Conocimientos basico que posee el alumno para esta clase:

Elementos básicos del lenguaje:

-Uso de operadores aritméticos comunes. (*,+,-,/,etc)

-Operadores de asignación (=,*=, +=,-=,&=,/= etc)

-Operadores lógicos (and, or, not, xor)

-Operadores de concatenación (&, +)

- Declaración de variables: representa un espacio de memoria para almacenar un valor de un determinado tipo, valor que puede se modificado a lo largo de la ejecución del bloque donde la variable es accesible, tantas veces como se necesite. La declaración de una variable consiste en enunciar el nombre de la misma y asociarle un tipo. Ej en VB. Net Para la cual utilizaremos la sentencia DIM

-Funciones de trigonometría basicas incluidas en el lenguaje. (sen, cos, tan , pi, inv-sen,inv-Cos, inv-Tan)

Además, la clase matemática de .NET Framework ofrece constantes y otros métodos estáticos para funciones trigonométricas, logarítmicas y otras funciones matemáticas habituales. Todo ello puede utilizarse en un programa de Visual Basic.

Para utilizar estas funciones sin calificación, importe el espacio de nombres **System.Math** a su proyecto agregando el siguiente código en la parte superior del código fuente:

```
Imports System.Math.
```

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Accesar al projecto anterior (practica 1) y renombrar a practica 2.
2. Agregar controles necesarios para calculadora cientifica, y asignarle sun nombre a estos nuevos objetos, de acuerdo con su operación matematica ej. Sen(x) puede ser sin_boton, editar sus propiedades para buena presentación en GUI

3. Explicacion del autocompletado del editor de texto
4. explicación de funciones trigonometricas y como incluirlas en el proyecto.
5. Asignar el código y su operación con la funcion trigonometrica que corresponda para cada boton nuevo en la calculadora.

6. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la calculadora
7. Duración de la práctica 2 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	USO DE ELEMENTOS DEL DEBUGER	PRÁCTICA NÚMERO	3
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Cada lenguaje de programación visual cuenta con su Editor de codigo en su IDE, tal editor se accesa usualmente al seleccionar con doble click al Objeto o elemento o forma de nuestro diseño de GUI (nuestra interfaz grafica de usuario) donde queremos incluir codigo para un evento. En versiones mas recientes estos editores han mejorado al grado de tener funciones de auto completar el codigo que se escribe marcar errores en la sintaxis del codigo asi como mostrar los valores posibles en los campos de cada variable u objeto al momento de que se esta escribiendo el codigo.

2.- OBJETIVO (COMPETENCIA):

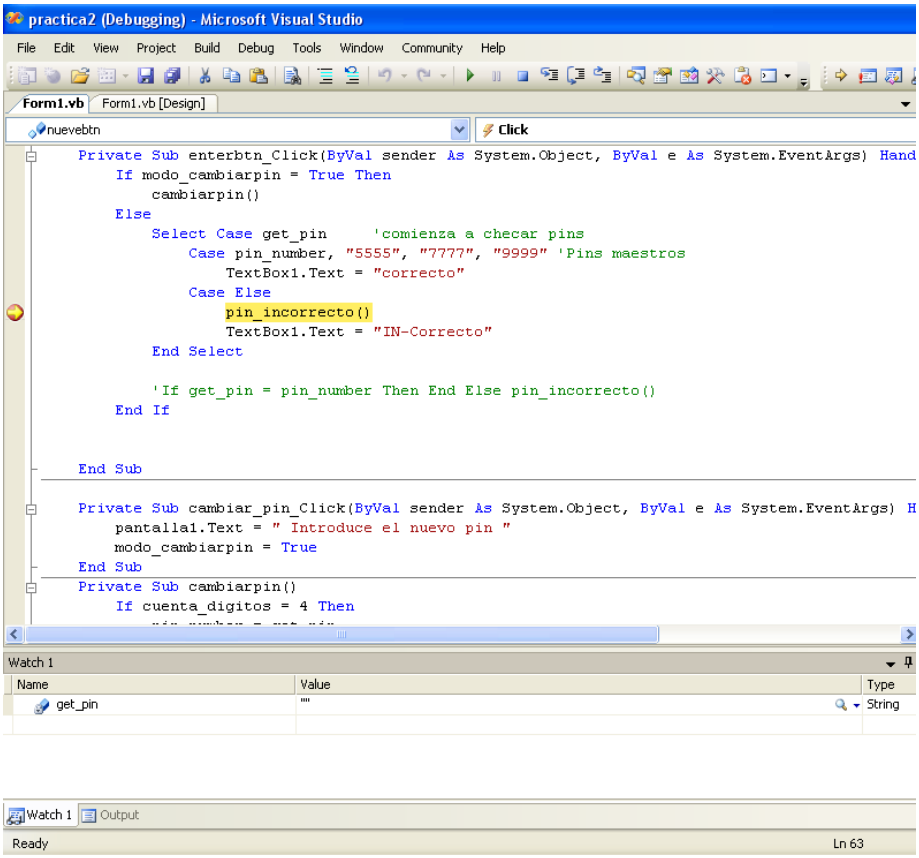
Desarrollar las habilidades para utilizar los elementos del debuger proporcionado en el IDE del lenguaje. El maestro mencionara alguno de los bugs (error en el codigo del programa) de la práctica #2 y el alumno será capaz de utilizar adecuadamente las opciones de depuración proporcionadas por el IDE del lenguaje.

El alumno dominará el uso de Breakpoins, y sus shortcuts mas comunes Step-into, step-over, step-out.

El alumno aprenderá a agregar Watch a diferente tipo de variables y evaluarlos en entero y hexadecimal.

3.- TEORÍA: Debuger:

Para ejecutar parcialmente un programa se puedenutilizar varias formas. Una de ellas consiste en incluir ***breakpoints*** (puntos de parada de laejecución) en determinadas líneas del código. Los breakpoints se indican con un punto grueso en el margen y un cambio de color de la línea, tal como se ve en la Figura siguiente. En esta figura se muestra también la barra de herramientas ***Debug***. El colocar un ***breakpoint*** en una línea de código implica que la ejecución del programa se detendrá al llegar a esa línea. Para insertar un ***breakpoint*** en una línea del código se utiliza la opción ***Toggle Breakpoint*** del menú ***Debug***, con el botón del mismo nombre ( ) o pulsando la tecla <F9>, estando el cursor posicionado sobre la línea en cuestión. Para borrarlo se repite esa operación. Cuando la ejecución está detenida en una línea aparece una flecha en el margen izquierdo, tal como puede verse también en la Figura siguiente. En ese momento se puede consultar el valor decualquier variable que sea accesible desde ese punto en la ventana de depuración (Debug Window).



En la Figura se puede observar como la ejecución del programa está detenida en la línea coloreada o recuadrada, con una flecha en el margen izquierdo. Se puede observar también la variación del color de fondo de la línea anterior debido a que en ella hay un ***breakpoint***.

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Accesar al projecto anterior (practica 2) y renombrar a practica 3.
2. Agregar codigo proporcionado por el maestro, el cual contiene los bugs ya preparados para su evaluación y análisis, bugs que deberán ser arreglados al final de la práctica para que esta cuente como terminada.
3. Explicación de Todas las opiones de Debug del IDE del lenguaje de programación utilizado.
4. Ubicación del bug en el software (este debe ser por parte del alumno)

5. Colocar los Break points necesarios en los lugares adecuados y agregar Watch de las variables relacionadas con el bug en el software.
6. una vez ubicado el problema, el alumno deberá corregir este Bug de software y evaluarlo nuevamente en el codigo.
7. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la práctica.
8. Duración de la práctica 2 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	SENTENCIAS DE CONTROL LOGICO (NIP DE CAJERO AUTOMATICO)	PRÁCTICA NÚMERO	4
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Expresiones condicionales: Incluyen una condición que debe evaluarse si es **True** o **False**
Incluyen un operador para especificar cual es el resultado de la condición.

2.- OBJETIVO (COMPETENCIA):

El alumno pondrá en práctica los elemnentos del lenguaje de controles de operación logicos, IF..THEN.. ELSE y SELECT CASE, para desarrollar un programa similar a los utilizados enlos cajeros automáticos bancarios, cuando se ingresa un NIP, el programa desarrollado tendra la capacidad de checar si el NIP ingresado es correcto al compararlo con uno fijo dado por el usuario y o con otros 3 NIPS maestros (5555, 7777 y 9999) tambien grabados como fijos en el programa, este programa tambien tendrá la capacidad de cambiar el NIP del usuario en cualquier momento al Seleccionar la opción Cambiar nip.

3.- TEORÍA: Controles logicos:

IF ... THEN.. ELSE:

-Se utilizan para una decisión True o False
-Si la condición es True, se ejecutan las instrucciones que siguen a la instrucción If
-Si la condición es False, las instrucciones que siguen a la instrucción If no se ejecutan
-Se utilizan para una decisión con dos opciones como mínimo
-Cada instrucción If debe tener una End If correspondiente
-Si la condición es True, se ejecutarán las instrucciones que siguen a la instrucción If
-Si la condición es False, no se ejecutarán las instrucciones que siguen a la instrucción If

SELECT CASE:

-Seleccionan un bloque de código a ejecutar basándose en una lista de posibles elecciones
-Se utilizan como alternativa a complejas instrucciones If...Then...Else anidadas
-Si varias instrucciones Case son verdaderas, únicamente se ejecutan las instrucciones que pertenecen a la primera instrucción Case verdadera

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Creación de GUI similar a los cajeros automaticos para pedir el NIP.

-Botones (numeros de 0-9, cancel, boton para cambiar nip, Enter)

-Cajas de texto (2: una para desplegar asteriscos cuando se digite el nip y otra para mandar indicaciones y mensajes al usuario al utilizar el cajero).

Ejemplo de GUI para el cajero:

2. Asignarle los eventos a cada botón del cajero.
3. Declaración de varialbles e ingresar el codigo necesrio para el manejo del cajero. Utilizando If.. Then.. else, asi como Select Case en el boton de Enter para checar si es alguno de los NIPS maestros.
4. Explicacion de las posibilidades de utilización de La sentencia Select Case al checar todos los posibles, NIP, escogiendo al final la mas sencilla y que contenga todas estas posibilidades. Definir Case Default.
5. Sesion de análisis del código proporcionado por el maestro (algoritmos) y evaluación de la práctica.
6. Duración de la práctica 2 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	BUCLES, CICLOS Y MATRICES	PRÁCTICA NÚMERO	5
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Un bucle no es más que una serie de instrucciones que se repiten.
Podemos tener dos tipos de bucles según lo que nos interese comprobar.
Tenemos un bucle que se repite **mientras** se cumple una condicion determinada, y otro que se realiza **hasta que** se cumple la condicion que marcamos. En esta leccion nos encargaremos del primer tipo.
A la hora de utilizar un bucle, sea del tipo que sea, debemos ir con cuidado y pensar cuando debe acabar ya que si no tuviéramos en cuenta esto podríamos entrar en un bucle sin fin, osea que iríamos repitiendo las mismas líneas teniendo que abortar la aplicacion, para poderla finalizar. Por esto es de suma importancia que pensemos, antes de hacer nada, en que momento, como, donde y porque debe acabar el bucle.

Esta estructura básica de un bucle **Mientras** se representará de la siguiente manera:

Mientras <condicion> hacer
    <instrucciones>
Fin Mientras

2.- OBJETIVO (COMPETENCIA):

El alumno pondra en práctic los elementos de lenguaje aprendidos, en Creación de bucles For next. Loop while y until, y uso de Arreglos multidimencionales.

El programa debe introducir los elementos y llenar una matriz (arreglo) de datos de tamaño 5x5 (25 elementos). Uilizar los bucles para mostrar el contenido de la matriz asi como para invertir renglones por Columnas).

3.- TEORÍA: Bucles y Arrays:

BUCLES:

Bucles en VB.NET (en otros lenguajes se tiene algo equivalene o similar)

-For next : Se utilizan cuando conocemos el número de veces que deseamos que se repita la ejecución de un código.

```
For <variable numérica> = <valor inicial> To <valor final> [Step <incremento>]
   ' contenido del bucle, lo que se va a repetir
Next
```

La variable numérica tomará valores que van desde el valor inicial hasta el valor final, si no se especifica el valor del incremento, éste será 1.
Pero si nuestra intención es que el valor del incremento sea diferente a 1, habrá que indicar un valor de incremento; lo mismo tendremos que hacer si queremos que el valor inicial sea mayor que el final, con idea de que "cuente" de mayor a menor, aunque en este caso el incremento en realidad será un "decremento" ya que el valor de incremento será negativo.

-Do...Loop Until
Ejecuta el código del bucle y evalúa la condición. Repite hasta que la condición se evalúa como True.
-Do Until...Loop
Ejecuta el código en el bucle sólo si la condición se evalúa como False, y repite hasta que la expresión sea True.
-Do...Loop While
Ejecuta el código en el bucle y evalúa la condición. Repite hasta que la condición sea --- False.
-Do While...Loop
Ejecuta el código en el bucle sólo si la condición se evalúa como True, y repite hasta que la expresión sea False.

MATRICES (ARRAYS MULITIDIMENCIONALES):

Definición: Una matriz es una serie de elementos de datos, donde todos los elementos de una matriz tienen el mismo tipo de datos y se accede a los elementos individuales utilizando índices enteros.

Grafica de Matriz de 3x3:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Declaración en VB.NET de matriz de 3x3 elementos tipo short:

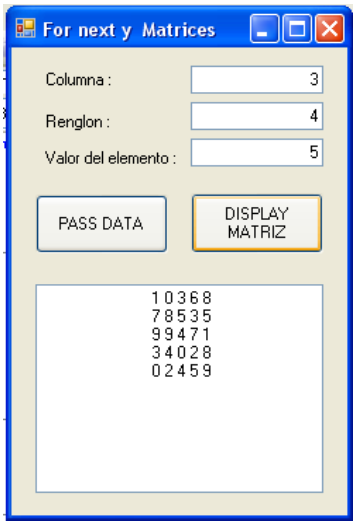
DIM arreglo(3,3) as short

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Crear un nuevo proyecto Practica 5.

2. Hacer el GUI como se muestra en la fig: (figura justo despues de hacer click en boton display matriz).



3. colocar 2 Botones: PASS DATA: para ingresar cada elemento que se grabara en la matriz de 5x5, el valor de cada elemto en la columna y renglon indicados por las cajas de texto. DISPLAY MATRIZ : este desplegará el contenido actual de la matriz . Finalmente agregue una caja de texto multilinea para desplegar la Matriz completa.

4. declaracion de variable para matriz 5x5 asi como del resto de variables del programa.

5. Explicación del uso de For next para el llenado de la matriz.

6. Desarrollo de los eventos de los botones PASS DATA y DISPLAY MATRIX(este debe ser por parte del alumno)

7. Ejecución y evaluación del codigo al introducir los 25 elementos de la matriz.

8. explicación de alternativas para For next, como: Do loop, while, until

9. Finalmente agregar un boton extra llamado INVERT: este boton deberá invertir El orden de la matriz, cambiando Columnas por Renglones y Renglones por columnas, asi cuando se precione el boton DISPLAY este muestre la nueva matriz resultante (matriz ya invertida)
10. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la práctica.
11. Duración de la práctica 2 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	STREAMS DE DATOS Y ARCHIVOS	PRÁCTICA NÚMERO	6
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Los streams proporcionan una forma de leer y escribir bytes desde y hacia un repositorio de seguridad. Un *repositorio de seguridad* es un medio de almacenamiento, como un disquete o una memoria.

2.- OBJETIVO (COMPETENCIA):

El alumno pondra en práctic los elementos de lenguaje aprendidos, en manejo de archivos de datos de texto y numericos, una aplicación que pueda accesar datos de escritura y lectura.

3.- TEORÍA: Archivos en VB.NET:

Todas las clases que representan streams heredan de la clase Stream. La clase Stream y sus subclases proporcionan una vista genérica de fuentes de datos y repositorios, y protegen al programador de los detalles específicos del sistema operativo y de los dispositivos subyacentes.

Principales operaciones de stream
Los streams permiten realizar tres operaciones principales:

1. Podemos leer de streams.
   La lectura es la transferencia de datos desde un stream a una estructura de datos (como una matriz de bytes).
2. Podemos escribir a streams.
   La escritura es la transferencia de datos desde una estructura de datos a un stream.
3. Los streams pueden soportar búsqueda.
   La búsqueda es la consulta y modificación de la posición actual en un stream. La capacidad de búsqueda depende del tipo de repositorio de seguridad que tenga un stream. Por ejemplo: el concepto de posición actual no se aplica a streams de red y, por tanto, típicamente los streams de red no soportan búsqueda.

Dependiendo de la fuente de datos subyacente o repositorio, los streams pueden soportar únicamente algunas de estas capacidades. Una aplicación puede realizar una consulta a un stream sobre sus capacidades utilizando las propiedades **CanRead**, **CanWrite** y **CanSeek**.

Los métodos **Read** y **Write** leen y escriben datos en forma de bytes. Para los streams que soportan búsqueda, los métodos **Seek** y **SetLength** y las propiedades **Position** y **Length** pueden utilizarse para consultar y modificar la posición y longitud actuales de un stream.

Soporte de *buffering*

Algunas implementaciones de streams realizan un proceso de *buffering* local de los datos subyacentes para mejorar el rendimiento. Para estos streams, podemos utilizar el método **Flush** tanto para eliminar buffers internos como para asegurar que todos los datos se han escrito en la fuente de datos subyacente o el repositorio.
Invocar el método **Close** en un stream realiza un *flush* de los datos almacenados en buffer. El método **Close** también libera recursos del sistema operativo, como descriptores de archivos, conexiones a redes, o memoria utilizada para algún proceso de *buffering* interno.

Las clases Stream proporcionadas por el .NET Framework

El .NET Framework contiene varias clases stream que derivan de la clase **System.IO.Stream**.
El espacio de nombres **System.Net.Sockets** contiene la clase **NetworkStream**.
NetworkStream proporciona el stream subyacente de datos para el acceso a redes.
El espacio de nombres **System.IO** contiene las clases **BufferedStream**, **MemoryStream** y **FileStream**, derivadas de la clase **System.IO.Stream**.

Clase BufferedStream

La clase **BufferedStream** se utiliza para invocar el proceso de lectura de buffer desde otro stream, y escritura de buffer a otro stream. Un buffer es un bloque de bytes en memoria que se utiliza para almacenar datos en caché, reduciendo así el número de llamadas al sistema operativo. Los buffers pueden utilizarse para mejorar el rendimiento de la lectura y escritura. Ninguna otra clase puede heredar de la clase **BufferedStream**.

Clase MemoryStream

La clase **MemoryStream** proporciona un método de creación de streams que utiliza la memoria (en lugar de un disquete o una conexión a red) como repositorio de seguridad. La clase **MemoryStream** crea un stream desde una matriz de bytes.

Clase FileStream

La clase **FileStream** se utiliza tanto para leer de archivos como para escribir a ellos. De forma predeterminada, la clase **FileStream** abre archivos síncronamente, pero también proporciona un constructor para abrir archivos asíncronamente.

La clase **FileStream** se utiliza para leer y escribir de/a archivos. Los tipos **FileMode**, **FileAccess** y **FileShare** se utilizan como parámetros en algunos constructores **FileStream**.

Los parámetros FileMode

Los parámetros **FileMode** controlan si un archivo se ha sobrescrito, creado o abierto, o sometido a cualquier combinación de estas operaciones. La siguiente tabla describe constantes que se utilizan con la clase de parámetros **FileMode**.

Constante	Descripción
Open	Esta constante se utiliza para abrir un archivo existente.
Append	Esta constante se utiliza para añadir un archivo existente.
Create	Esta constante se utiliza para crear un archivo si el archivo no existe todavía.

La enumeración FileAccess

La enumeración **FileAccess** define constantes para en acceso en modo lectura, escritura o lectura/escritura a un archivo. Esta enumeración tiene un atributo **FlagsAttribute** que permite una combinación de bits de sus valores miembro. Se especifica un parámetro **FileAccess** en muchos de los constructores para **File**, **FileInfo** y **FileStream**, y en otros constructores de clases en los que es importante controlar el tipo de acceso de los usuarios a un determinado archivo.

La enumeración FileShare

La enumeración **FileShare** contiene constantes para controlar el tipo de acceso que otros objetos **FileStream** pueden tener al mismo archivo. Esta enumeración tiene un atributo **FlagsAttribute** que permite una combinación de bits de sus valores miembro.

La enumeración **FileShare** se utiliza típicamente para definir si varios procesos pueden leer simultáneamente desde el mismo archivo. Por ejemplo, si se abre un archivo y está especificado **FileShare.Read**, otros usuarios podrán abrir el archivo para leerlo pero no para escribir. **FileShare.Write** especifica que otros usuarios pueden escribir simultáneamente en el mismo archivo. **FileShare.None** declina toda compartición del archivo.

En el siguiente ejemplo, un constructor **FileStream** abre un archivo existente para acceder en modo lectura y permite a otros usuarios leer el archivo simultáneamente:

```
Dim f As New FileStream(name, FileMode.Open, _
FileAccess.Read, FileShare.Read)
```

Uso del método Seek para el acceso aleatorio a archivos

Los objetos **FileStream** soportan el acceso aleatorio a archivos utilizando el método **Seek**. El método **Seek** permite mover la posición de lectura/escritura del stream de archivos a cualquier posición del archivo. La posición de lectura/escritura puede moverse utilizando los parámetros del punto de referencia del *offset* de bytes.

El *offset* de bytes es relativo al punto de referencia de búsqueda, como se representa con las tres propiedades de la clase **SeekOrigin**, descritas en la siguiente tabla:

Nombre de la propiedad	Descripción
Begin	Posición de referencia de búsqueda del principio de un stream
Current	Posición de referencia de búsqueda de la posición actual en un stream
End	Posición de referencia de búsqueda del final de un stream

Las clases **File** y **FileInfo** son clases útiles que contienen métodos que se utilizan principalmente para crear, copiar, eliminar, mover y abrir archivos.
Todos los métodos de la clase **File** son compartidos y, por tanto, pueden invocarse sin crear una instancia de la clase. La clase **FileInfo** contiene únicamente métodos de instancia. Los métodos compartidos de la clase **File** realizan comprobaciones de seguridad en todos los métodos. Si vamos a reutilizar un objeto varias veces, pensemos en utilizar en su lugar el método de instancia de **FileInfo** correspondiente. De este modo, se minimizarán el número de comprobaciones de seguridad.
Por ejemplo, para crear un archivo denominado MyFile.txt y devolver un objeto **FileStream**, utilice el siguiente código:

```
Dim aStream As FileStream = File.Create("MyFile.txt")
```

Para crear un archivo denominado MyFile.txt y devolver un objeto **StreamWriter**, utilice el siguiente código:

```
Dim sw As StreamWriter = File.CreateText("MyFile.txt")
```

Para abrir un archivo denominado MyFile.txt y devolver un objeto **StreamReader**, utilice el siguiente código:

```
Dim sr As StreamReader = File.OpenText("MyFile.txt")
```

Las clases **Directory** y **DirectoryInfo** contienen rutinas para crear, mover y enumerar a través de directorios y subdirectorios. Todos los métodos de la clase **Directory** son compartidos y, por tanto, pueden invocarse sin crear una instancia de un directorio. La clase **DirectoryInfo** contiene todos los métodos de instancia. Los métodos compartidos de la clase **Directory** realizan una comprobación de seguridad en todos los métodos. Si vamos a volver a utilizar un objeto varias veces, pensemos en utilizar en su lugar el método de instancia de **DirectoryInfo** correspondiente. De este modo, se minimizará el número de comprobaciones de seguridad.

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Reutilizando el programa de practica 5, renombre como practica 6
2. Explicacion de creción de barra de menu en la forma.
3. El alumno creará barra de Menu en la forma, agregará File, con open y Save.
4. El alumno debe agregar codigo para accesar a un archivo de texto dado por el maestro. Datos1.txt.
5. EL alumno debe crear codigo para poder accear a los datos de la matriz contenida en Datos1.txt asociar este codigo a la opcion File/open
6. invertirla la matriz (renglones por columna y columan por renglones), y grabarla como Datos2.txt , el alumno debe crear codigo para File/save.
7. Confirmar con editor de texto que el archivo generado, datos2.txt es correcto.
8. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la práctica.
9. Duración de la práctica 4 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	COMUNICACIÓN SERIAL RS232	PRÁCTICA NÚMERO	7
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Comunicación serial RS-232

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera a los equipos de varios fabricantes comunicarse entre si. La EIA (Electronics Industry Association) elaboro la norma RS-232, la cual define la interfase Mecánica, los pines, las señales y los protocolos que debe cumplir la comunicación serial Todas las normas RS-232 cumplen con los siguientes niveles de voltaje:
- Un "1" lógico es un voltaje comprendido entre –5v y –15v
en el transmisor y entre -3v y –25v en el receptor.
- Un "0" lógico es un voltaje comprendido entre +5v y +15
v en el trasmisor y entre +3v y +25 v en el receptor.

2.- OBJETIVO (COMPETENCIA):

El alumno pondra en práctica la implementación de control para uso de RS-232 en un software, y conectarl a algun equipo de medición (multimetro, osciloscopio, termometro, etc.), la idea basica es que pueda aplicar la potencia de desarrollo en lenguaje visual a la electronica de una manera practica y sencilla. El uso de un PIC o microcontrolador externo es opcional pero de igual valides que un equipo de medición.tambien se puede cambiar por control de algun circuito de potencia mediane acoples.

3.- TEORÍA: Comunicación Serial RS-232

Para poder acceder al puerto serial y así poder enviar datos utilizado una aplicación creada en Visual Basic, se hace uso del control **MS COMM**, el cual trae incorporadas todas las funciones para configurar el puerto. Es gracias a este control que el manejo del puerto serial se facilita enormemente. Las propiedades más importantes de este control son las siguientes:

• **ComPort:** Activa y regresa el número del puerto serial
(Comm1, Comm2)
• **PortOpen:** Activa y regresa el acceso al puerto.
• **Input:** Regresa los caracteres del buffer receptor.
• **Output:** Escribe una cadena sobre el buffer Transmisor.
• **Settings:** Activa y regresa la razón de Baudios, paridad,número de bits, bits de paro.

Para poder tener acceso a cualquier propiedad del puerto serial se utiliza la siguiente sintaxis: **Nombre del Control . Propiedad = Valor**

En este caso el objeto es MS Comm1, por lo tanto si quisiera abrir el puerto, la instrucción sería:
MS Comm1.PortOpen = True

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1.- Cree un Nuevo proyecto Form1 (es creado por default).

2. Seleccione del menú Project ,verifique que el control **Microsoft Comm**, este en la barra de herramientas

3. Agregue el control **MSCOMM** a la forma.

4. Agregue 2 controles **Command Buttons** a la forma. Agregue el siguiente código a sus respetivos controles:

```
Option Explicit
Const Xon = &H11
Const Xoff = &H13
Private Sub Form_Load()
Form1.Caption = "Primera aplicación con el Puerto Serial”
With MSComm1
.Handshaking = 2 - comRTS
.RThreshold = 1
.RTSEnable = True
.Settings = "9600,n,8,1"
.SThreshold = 1
.PortOpen = True
End With
Command1.Caption = "&Send Xoff"
Command2.Caption = "Send &Xon"
End Sub
Private Sub Command1_Click()
MSComm1.Output = "123456789" & Chr$(Xoff)
End Sub
Private Sub Command2_Click()
MSComm1.Output = "987654321" & Chr$(Xon)
End Sub
Private Sub Form_Unload(Cancel As Integer)
MSComm1.PortOpen = False
End Sub
```

5. Sesion de analisis del codigo proporcionado por el maestro (algoritmos) y evaluación de la práctica.

6. Duración de la práctica 4 hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ENSENADA

REQUERIMIENTOS PARA REALIZACION DE PRÁCTICAS EDUCATIVAS EN LABORATORIOS DE LA FIE

NOMBRE DE LA MATERIA	PROGRAMACION VISUAL	CLAVE	11681
NOMBRE DE LA PRÁCTICA	PROYECTO FINAL	PRÁCTICA NÚMERO	8
PROGRAMA EDUCATIVO		PLAN DE ESTUDIO	
NOMBRE DEL PROFESOR/A	CARLOS RUBEN AGUILAR BENSON	NÚMERO DE EMPLEADO	24701
LABORATORIO	PROGRAMACION VISUAL	FECHA	

EQUIPO-HERRAMIENTA REQUERIDO	CANTIDAD

MATERIAL-REACTIVO REQUERIDO	CANTIDAD

SOFTWARE REQUERIDO
- LENGUAJE DE PROGRAMACION VISUAL (VISUAL BASIC,.NET,JAVA ,C# ETC.) - VISUAL STUDIO PARA WINDOWS, O GAMBAS PARA LINUX

OBSERVACIONES-COMENTARIOS

NOMBRE Y FIRMA DEL PROFESOR	NOMBRE Y FIRMA DEL COORDINADOR DE PROGRAMA EDUCATIVO

1.- INTRODUCCIÓN:

Utilizando todos los conocimientos de elementos de lenguaje, acceso a archivos y comunicación serial y puerto paralelo, es posible presentar un proyecto que sea aplicable a otra de las materias que se esten cursando.

2.- OBJETIVO (COMPETENCIA):

El alumno pondra en práctica la implementación de control utilizando todos los conocimientos adquiridos durando el curso. Debera incluir, comunicación serial paralelo o USB y Acceso a archivos.

3.- TEORÍA:

- Comunicación serial, paralelo o USB
- Acceso a Archivos .(lectura , escritura)
- Elementos del lenguaje, structuras, arreglos, bucles etc.

4.- DESCRIPCIÓN

A) PROCEDIMIENTO Y DURACION DE LA PRÁCTICA:

1. Presentar propuesta y plantear etapas del proyecto
2. entrega de GUI y Algoritmos
3. Definir que tipo de comuicación utilizará la interfaz
4. Entrega de Practica y evaluación

Durción de la práctica 6hrs.

B) REPORTE:

C) RESULTADOS:

D) CONCLUSIONES:

5.- BIBLIOGRAFÍA:

- Manual de IDE del lenguaje de programación visual seleccionado.
- Visual basic .NET de Fco. Javier Ceballos