Versione N° 1.0

LaurTec

Progetto di un'interfaccia grafica per il controllo della porta seriale

Autore : *Marcello Pinna*

ID: UT0001-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione

In rete sono presenti molte risorse che spiegano in modo dettagliato come creare, nell'ambiente di sviluppo .NET Visual Basic, un'interfaccia grafica (GUI) da utilizzare per comunicazioni di tipo seriale. Poiché sono un autodidatta e tutte le mie conoscenze le ho studiate e sperimentate sulla mia pelle, ho deciso di scrivere ugualmente questo tutorial per cercare di mettere nelle giuste condizioni chiunque abbia voglia di creare una propria applicazione utilizzando, appunto, questo linguaggio. Nella stesura del testo non tratterò la teoria, non parlerò di classi, di ereditarietà di classi o di gerarchia: per chi ha voglia di approfondire la propria conoscenza personale rimando alle proficue letture che si possono trovare sul sito MSDN della Microsoft. Procederò, invece, passo dopo passo, in modo che tutti siano in grado di ripetere ciò che ho fatto. A coloro che si avvicinano per la prima volta a questo mondo, consiglio di scaricare il manuale "A scuola con Visual Basic", reperibile presso questo indirizzo internet:

http://www.vbscuola.it/VB2010/A_Scuola_con_VB2010.pdf

E' un manuale di oltre 1000 pagine ma scorrevole alla lettura, dove si troveranno molte indicazioni utili sull'installazione del software e molti esercizi con i quali si può iniziare realmente a programmare. Se non tutto, invito i neofiti a studiarne con attenzione almeno le prime tre parti, con relativi capitoli. Partirò, quindi, dal presupposto che l'utente sappia già muoversi, con una certa disinvoltura, dentro i meandri di Visual Basic 2010 Express e che conosca, almeno a grandi linee, cosa sia una comunicazione seriale.

Un po' di ragionamento logico...

Nel caso specifico della progettazione della nostra interfaccia le domande essenziali da porsi sono:

- Cosa devo prevedere e fare affinché il mio PC possa comunicare con il mio hardware?
- Che tipo di dati voglio trasmettere e ricevere?
- Come li trasmetto?
- Dove li ricevo?
- Quanto tempo è necessario per la trasmissione e per la ricezione?

La risposta alla prima domanda è: impostare innanzitutto i parametri di comunicazione tra i due sistemi, che devono essere tra loro identici.

Possibili risposte alla seconda, terza, quarta e quinta domanda sono rispettivamente: qualunque tipo di dato, un buffer di trasmissione gestito da un pulsante, un buffer di ricezione, i quali possono essere impostati a nostra discrezione.

Per cui, nel progetto, dovranno essere previste cinque sezioni (o contenitori), rappresentate da cinque controlli chiamati GroupBox, all'interno dei quali sistemeremo gli altri controlli necessari. Chiameremo, rispettivamente, queste sezioni:

- RS232 Setting
- Options
- TX buffer
- RX buffer
- Delay ms

Naturalmente ognuno potrà scegliere i nomi che vuole ma io consiglio dei nomi intuitivi.

Ora analizziamo queste sezioni una per una.

L'impostazione della porta seriale prevede almeno cinque parametri che sono:

- Il nome della porta COM utilizzata.
- La velocità di comunicazione espressa in baudrate (bit al secondo).
- La parità.
- Il formato (databits).
- Il bit di stop (stopbit).

Quindi dovremo prevedere ulteriori cinque controlli che, all'interno del groupbox "RS232 setting" ci permettano di scegliere indipendentemente le varie impostazioni di comunicazione.
Stessa cosa dovremo fare per il groupbox "Options" per il tipo di dato da inviare, che può essere del seguente tipo:

- Testo
- Byte
- Una conversione carattere-intero
- Una conversione intero-carattere

In questo caso ci serviranno quattro controlli aggiuntivi. Stessa logica seguiremo per il resto delle sezioni, per cui avremo un controllo per la trasmissione, uno per la ricezione ed due per gestire i ritardi.

I controlli scelti per questa applicazione sono: cinque ComboBox per RS232 setting, quattro RadioButton per Options, una TextBox per TX buffer, una RichTextBox per RX buffer, due TextBox per Delay ms ed un Button per gestire il buffer in trasmissione.

Perché la scelta di questi controlli?

- Il controllo combobox perché ha la grande peculiarità di poter essere popolato da un insieme di elementi ovvero, detto volgarmente, si può inserire al suo interno una lista (elenco) di opzioni (ad esempio una lista di porte COM presenti nel PC, oppure ad una lista di valori di baudrate ecc.).
- Il controllo radiobutton perché, se uno di essi è attivo, esclude mutuamente tutti gli altri controlli ad esso identici.
- Il controllo textbox perché è il più idoneo alla scrittura e quindi, all'invio dati.
- Il controllo richtextbox perché può essere formattato sia in scrittura che in lettura secondo la tipologia di dato trattato.
- Il controllo button come sostituzione del tasto invio.

Ma tutto questo non basta se manca il controllo più importante attorno al quale ruota tutto il sistema, ovvero il controllo SerialPort, disponibile nell'ambiente di sviluppo .NET. Inoltre, dovremo prevedere altri pulsanti per la connessione, la disconnessione, la chiusura dell'interfaccia grafica, la possibilità di salvare e poi ricaricare le impostazioni di comunicazione definite dall'utente e la possibilità di visualizzare lo stato di connessione dell'interfaccia stessa. Per realizzare tutto questo ci serviranno altri controlli aggiuntivi chiamati MenuStrip e StatusStrip. Ricordo che tutte le modifiche sui controlli

verranno effettuate nella finestra Proprietà e che, al fine di agevolare la scrittura e la lettura del codice, alcuni di essi, i più importanti, verranno identificati da un prefisso. Una Tabella riepilogativa che li riassume tutti è la seguente:

n°	Controllo	Prefisso	Proprietà	
5	GroupBox		Text	
5	ComboBox	cbo	Name	DropDownStyle
4	RadioButton		Text	
3	TextBox	txt	Name	Multiline
1	RichTextBox	rtb	Name	ReadOnly
1	Button	btn	Name	Text
7	Label		Text	
2	StatusLabel	lbl	Name	
1	MenuStrip		Text	
1	StatusStrip			
1	SerialPort			
1	SaveFileDialog			
1	OpenFileDialog			

Tabella 1: *Riassunto dei controlli da aggiungere.*

Hardware necessario

- PC con sistema operativo XP o superiore, dotato di porta COM o adattatore USB-RS232.
- Visual Basic 10 Express, installato nel PC secondo le modalità previste.
- Cavo seriale DB9.
- Scheda di sviluppo Freedom I o Freedom II, reperibili presso il sito www.LaurTec.it .
- Microcontrollore PIC18F4550.

Sviluppo dell'interfaccia Grafica

Avviamo VB10 Express, nella pagina iniziale del programma andiamo su File, Nuovo Progetto e nella finestra del menù dei modelli selezioniamo Applicazione Windows Form.
Assegniamo un nome all'applicazione (es. SerialPort) e premiamo OK. Quando ci appare la finestra di progettazione Form1, facciamo un click su File e poi su Salva tutto. Apparirà una finestra con la richiesta di salvare l'intero progetto, a questo punto premiamo il pulsante Salva. L'intero progetto verrà salvato in una cartella all'interno della directory "Visual Studio 2010\Projects" (che si trova normalmente sotto la directory Documenti). Facciamo un click su Form1, andiamo nella finestra delle proprietà, selezioniamo la voce Text e cambiamo il testo presente con SerialPort, assegnando così un titolo alla nostra interfaccia. Poi impostiamo la proprietà MaximizeBox su False

Ora allarghiamo il Form1 alle dimensioni da noi previste ed inseriamo cinque controlli: MenuStrip, StatusStrip, SerialPort, SaveFileDialog, OpenFileDialog (l'ordine non ha importanza). Il risultato sarà simile a quello della Figura 1.

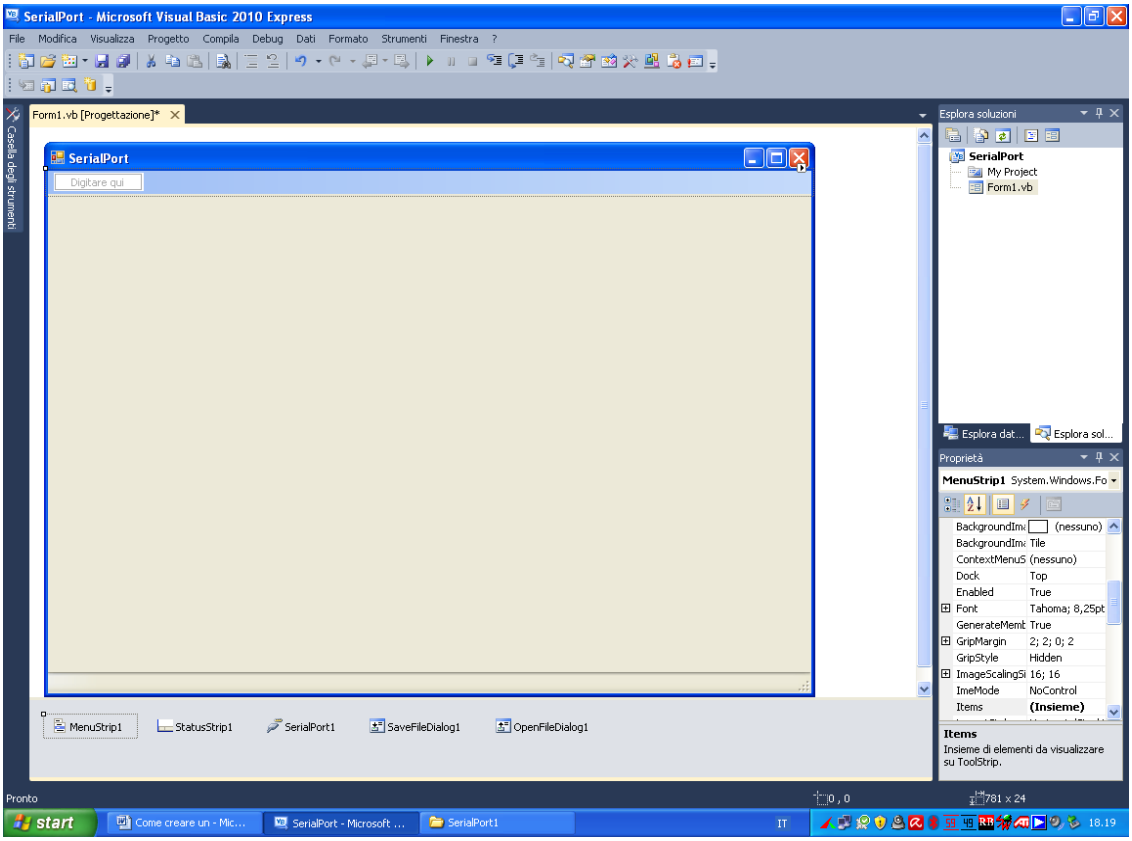

Figura 1: *Esempio del progetto con i relativi controlli.*

Ci posizioniamo in alto, sul controllo MenuStrip e, facendo un click sulla casella "Digitare qui" inseriamo, spostandoci in basso o a destra, tutte le voci dei menù e sotto menù che ci interessano secondo il seguente schema (in alternativa, possiamo usare la proprietà Text):

Menù	**Sotto menù**	
File	Load Settings	(serve a caricare le impostazioni precedentemente salvate)
	Save Settings	(serve a salvare le impostazioni scelte da'utente)
	(separatore)	
	Exit	(esce dal programma)
RS232	Connect	(connette il sistema)
	Disconnect	(disconnette il sistema)
	(separatore)	
	Clear Buffer	(pulisce il buffer RX)

Il risultato sarà simile a quello di Figura 2 e Figura 3.

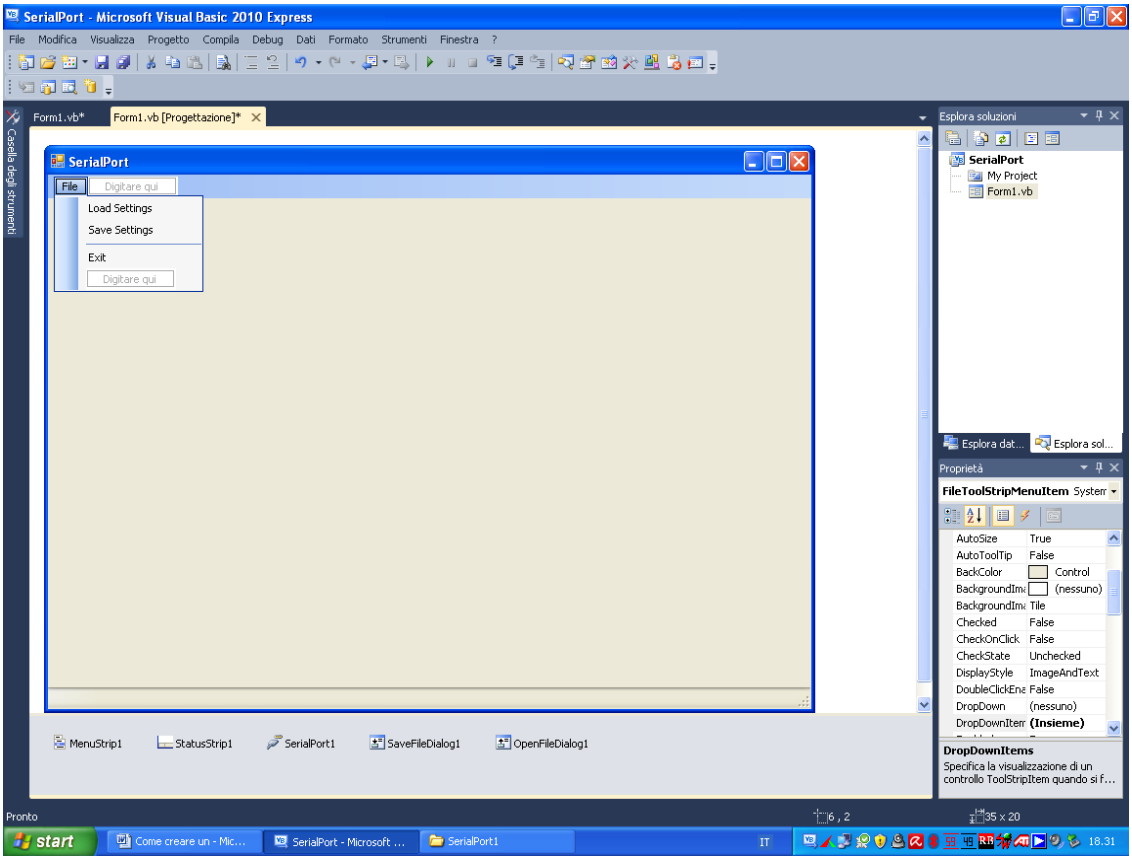

Figura 2: *Esempio di menu File..*

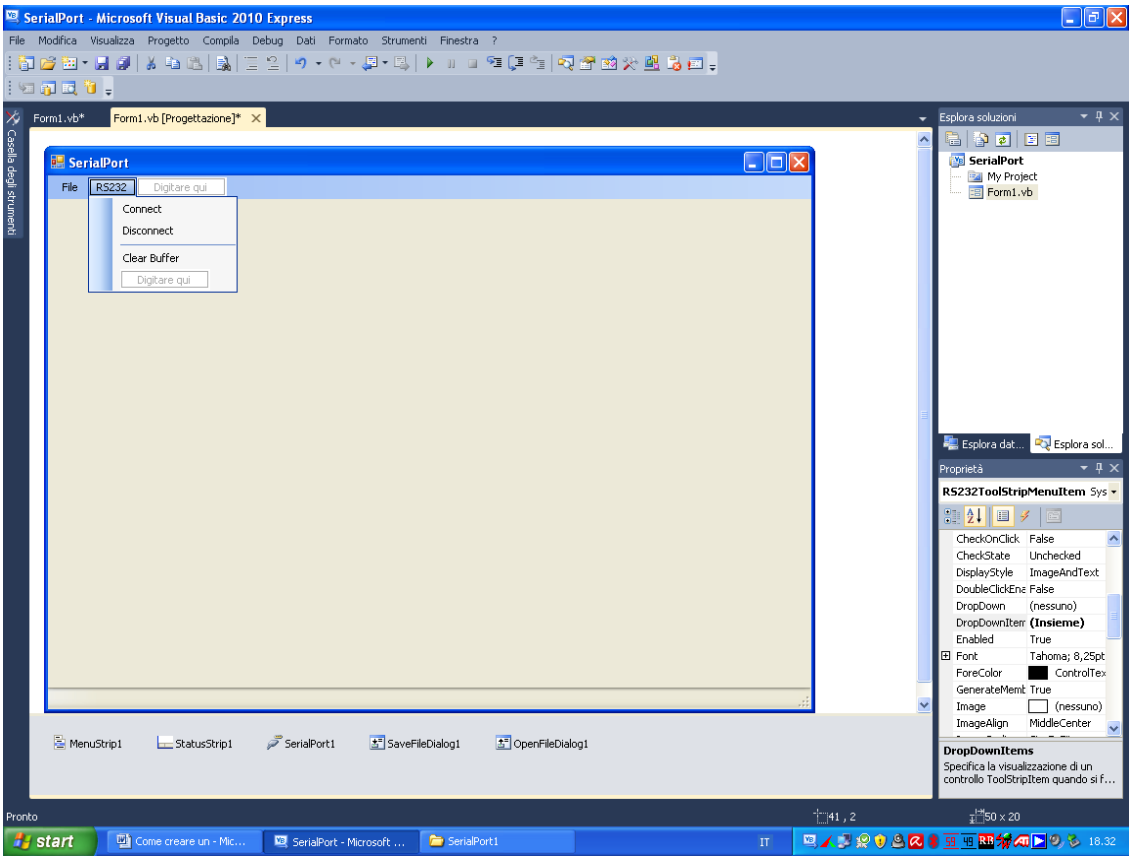

Figura 3: *Esempio di menu RS232..*

Facciamo un click in basso sul Form1, appena sopra l'area grigia, in modo da evidenziare il controllo StatusStrip, facciamo un click nel menù a tendina che ci appare a sinistra del controllo stesso e selezioniamo due StatusLabel. Tenendo presente la Tabella 1, rinominiamo la prima come lblSetting e la seconda come lblStatus, con prefisso lbl. Quest'ultima deve essere posizionata adiacente alla prima, come in Figura 4.

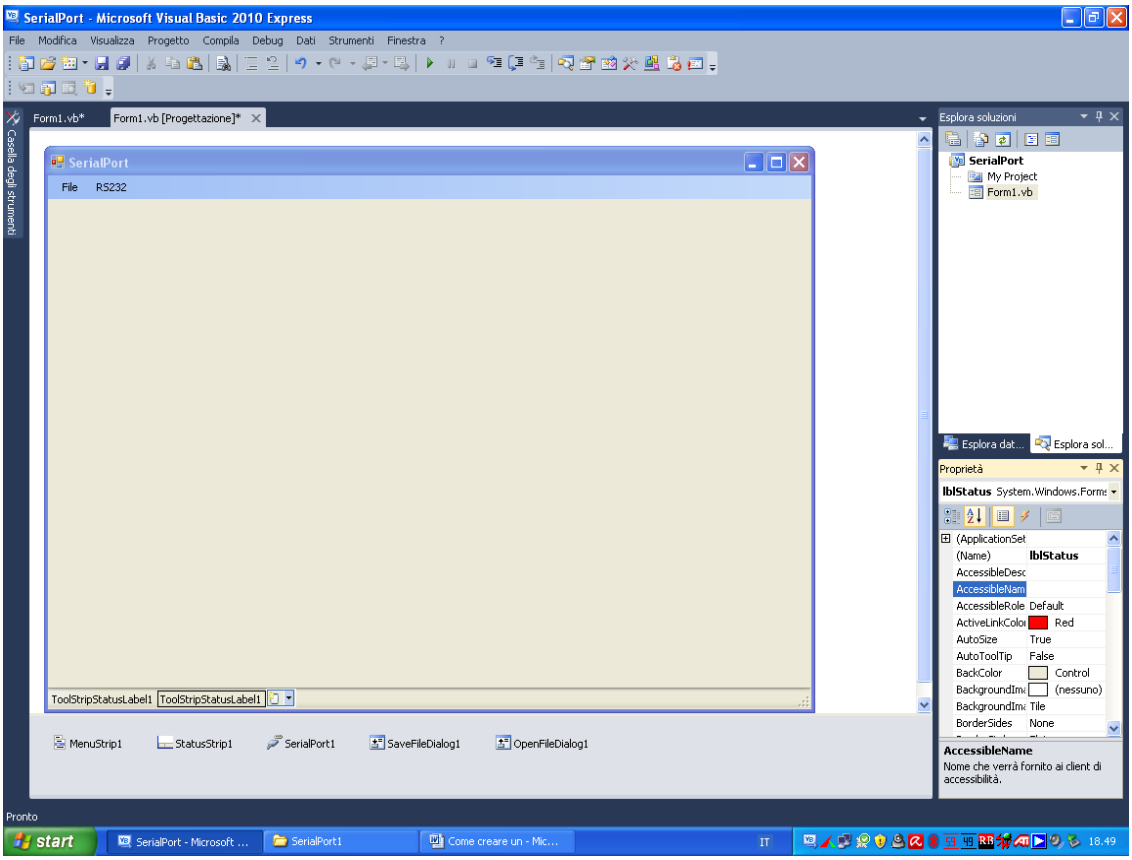

Figura 4: *Nomenclatura dei controlli e posizione delle etichette.*

Ora inseriamo cinque controlli groupbox e li posizioniamo secondo il nostro gusto estetico personale. Per ognuno di essi selezioniamo la voce Text nella finestra delle proprietà e li rinominiamo come:

- RX buffer
- TX buffer
- RS232 setting
- Delay ms
- Options

Per il tipo di carattere, lo stile e la grandezza selezionare Font dalla finestra delle proprietà, per il colore selezionare ForeColor.

A questo punto, tenendo sempre presente la Tabella 1, iniziamo ad inserire tutti i controlli mancanti.

All'interno del groupbox "RX buffer" inseriamo un controllo RichTextBox, che rinominiamo (con prefisso) come rtbDisplay, ci posizioniamo sulla proprietà ReadOnly e la impostiamo su True.

All'interno del groupbox "TX buffer" inseriamo un controllo TextBox, che rinominiamo (con prefisso) come txtSend e verifichiamo che la proprietà Multiline sia impostata su False.

All'interno del groupbox "RS232 setting" inseriamo ed accodiamo prima cinque controlli Label, che rinominiamo rispettivamente come:

- Port
- Baud
- Parity
- Format
- Stop Bit

Successivamente inseriamo cinque controlli ComboBox che rinominiamo (con prefisso) rispettivamente come:

- cboPort
- cboBaudRate
- cboParity
- cboFormat
- cboStopbit

e li accodiamo affiancati ai controlli Label. Ancora, per ognuno di essi, nella finestra Proprietà, impostiamo la voce DropDownStyle su DropDownList.

Ora inseriamo due controlli Label e due controlli TextBox all'interno del groupbox "Delay ms" e li allineiamo alternati. Rinominiamo le label rispettivamente come:

- Tx
- Rx

Rinominare le textbox (con prefisso) in:

- txtDelayTX
- txtDelayRx

Anche in questo caso verifichiamo che la proprietà Multiline sia impostata su False.

Inseriamo quattro controlli RadioButton nel groupBox "Options", li accodiamo e li rinominiamo rispettivamente come:

- Write Text
- Write Bytes
- Convert Chat to Int
- Convert Int to Char

Ora inseriamo un controllo Button, che servirà per la trasmissione dei dati, lo rinominiamo (con prefisso) come btnSend e nella proprietà Text scriviamo Send Data. Il risultato finale sarà simile a quello di Figura 5.

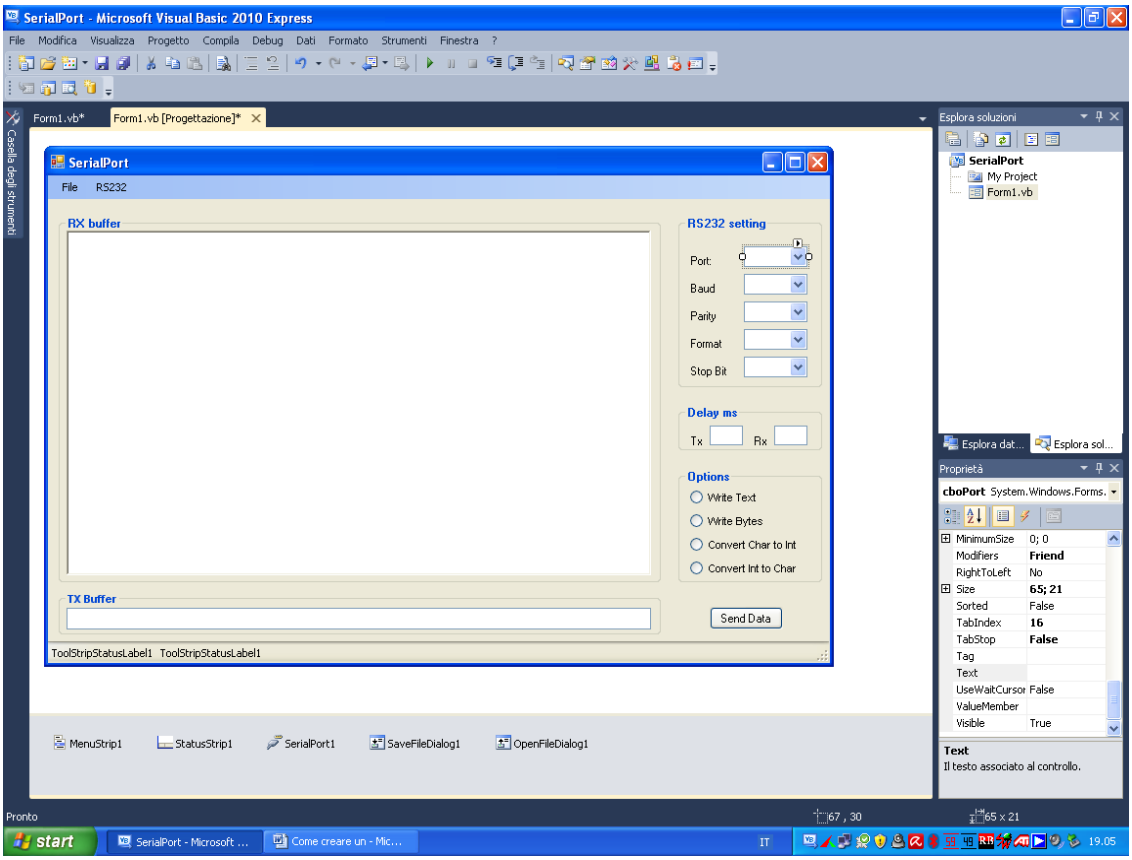

Figura 5: *Esempio grafico dopo l'inserimento dei vari controlli.*

A questo punto, abbiamo terminato la fase di progettazione dell'interfaccia per cui salviamo l'intero progetto andando su File e successivamente su Salva Tutto.

Ora possiamo concentrarci sulla scrittura del codice.

Alcune considerazioni sul codice

Nella scrittura di un codice serve un po' di fantasia, dobbiamo infatti immaginare quali eventi devono succedersi dall'avvio dell'interfaccia, alla sua chiusura e durante il suo utilizzo. Un esempio potrebbe essere questo: voglio o no che, al suo avvio, tutti i combobox siano popolati (riempiti) da una lista? La scelta è quindi a discrezione mia, quindi del progettista che dovrà decidere come impostare i vari eventi. Personalmente mi piace semplificare, perché evito errori non voluti.
Immaginiamo che decida di non popolare i combobox ma di scrivere direttamente all'interno di essi i miei parametri di comunicazione. Se il mio valore di baudrate scelto per lavorare è 19200 ma io scrivo per errore 19100 o 19300 ecco che ho un problema. Se non me ne accorgo sono guai, tradotti in una perdita di tempo. Per evitarli, meglio semplificare. Ecco perché ho immaginato un'interfaccia semplice, ma funzionante, che in linea generale:

All'avvio

- Abilitasse e popolasse tutti i combobox, con scrittura disabilitata.
- Abilitasse i textbox che gestiscono i ritardi.
- Abilitasse di default il radiobutton per l'invio dei dati in formato testo.

- Avesse il buffer di trasmissione disabilitato.
- Avesse il pulsante di disconnessione disabilitato.

Durante la connessione

- Disabilitasse tutti i combobox.
- Disabilitasse i textbox che gestiscono i ritardi.
- Permettesse di cambiare il tipo di dato da inviare .
- Abilitasse il buffer di trasmissione.
- Potesse inviare un numero di digit a mia discrezione, secondo il tipo di dato trattato.
- Avesse il pulsante di connessione disabilitato.

Durante la disconnessione

- Riportasse tutto alle condizioni di lavoro iniziali.

Come è strutturato il codice

Il codice è suddiviso in "regioni", ovvero zone all'interno delle quali vengono raccolte tutte le parti del sorgente che appartengono ad una categoria o ad uno stesso controllo. Se ho previsto diverse variabili le raccoglierò tutte in un'unica regione. Se ho previsto cinque combobox raccoglierò le loro singole routine in un'unica regione. Questo serve, oltre ad avere un codice molto ordinato, a permettere al programmatore di effettuare qualunque modifica senza leggerlo completamente. Ogni regione inizia con "#region" e termina con "#end region". E' possibile scrivere dei commenti riassuntivi per eventuali descrizioni degli eventi. L'ordine con cui vengono elencate le regioni non ha nessuna influenza sull'esecuzione corretta del programma, per cui sarà compito del programmatore sistemarle come meglio preferisce.

Gli Eventi nel codice

Esiste però un problema di fondo: non basta fare un semplice copia-incolla del codice completo per far funzionare il programma. Per alcuni controlli è necessario prima creare degli "eventi" e poi scrivere le specifiche routine. Per esempio, l'aggiornamento della label "lblSettings" avviene ogni volta che si avvia l'interfaccia o si seleziona una porta COM diversa. Questa selezione provoca l'evento SelectedIndexChanged che, a sua volta, lancia la routine Update_Setting. Per creare questo evento è sufficiente andare nella finestra Progettazione (Form1), selezionare il combobox cboPort, andare nella finestra delle proprietà, premere il pulsante con l'icona "fulmine" (Eventi), selezionare l'evento SelectedIndexChanged e fare un doppio click. Solo a quel punto potremo scrivere o copiare la parte di codice che ci interessa. Riassumendo: per ogni controllo per cui è previsto un evento, prima dovremo crearlo e solo dopo inserire la parte di codice che ci interessa, il resto può essere copiato senza problemi. I controlli interessati alla creazione di questi eventi sono riassunti nella Tabella 2:

Controllo	Evento
Load Settings	Click
Save Settings	Click
Exit	Click
Connect	Click
Disconnect	Click
cboPort	SelectedIndexChanged
cboBaudrate	SelectedIndexChanged
cboParity	SelectedIndexChanged
cboFormat	SelectedIndexChanged
cboStopbit	SelectedIndexChanged
RadioButton1	CheckedChanged
RadioButton2	CheckedChanged
RadioButton3	CheckedChanged
RadioButton4	CheckedChanged
btnSend	Click

Tabella 2: *Tipologia di eventi utilizzati per i vari controlli.*

L'esempio di quanto sopra descritto può essere visto in Figura 6, con la parte di codice (evidenziato in giallo) inserito all'interno dell'evento appena creato per il combobox "cboPort".

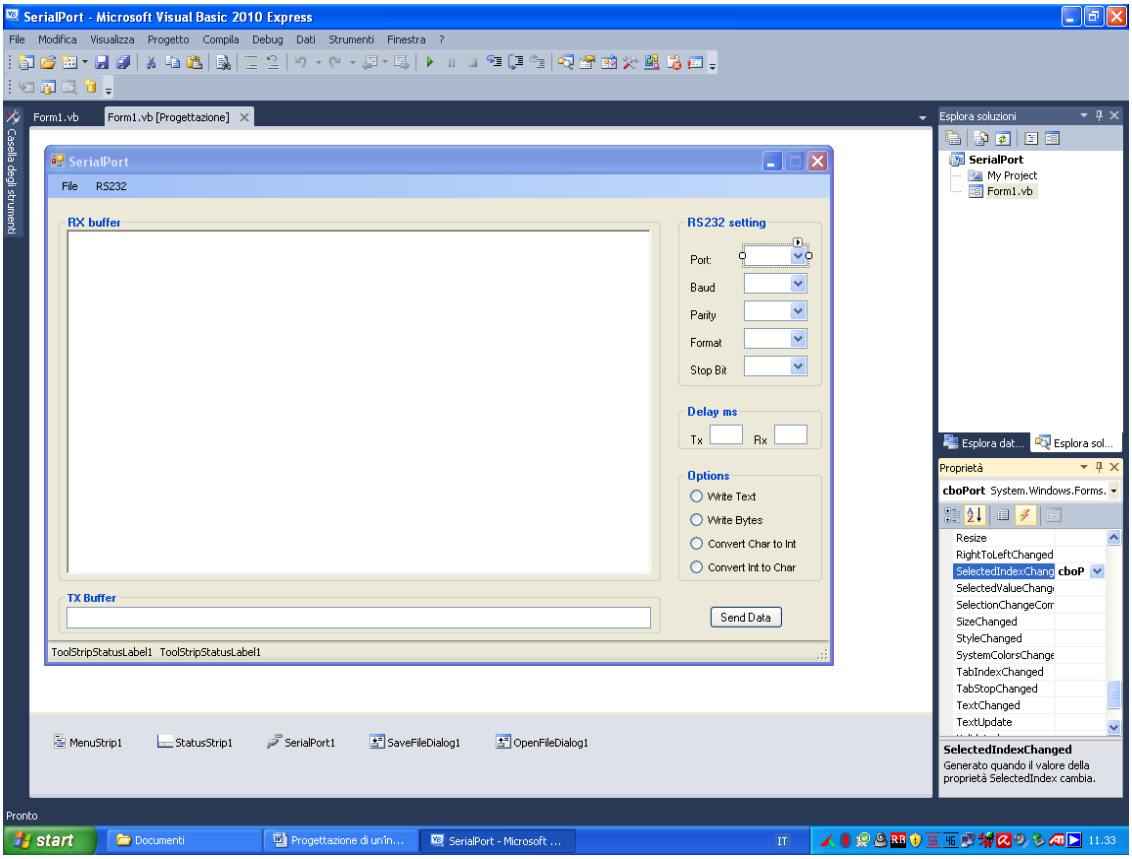

Figura 6: *Esempio di gestione dell'evento Index Changed.*

```
Private Sub cboPort_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cboPort.SelectedIndexChanged
        Update_Setting()
    End Sub
```

Codice completo

Riporto qui di seguito il codice completo, confidando nella sua comprensione da parte degli utenti. Il progetto completo può essere scaricato dal sito www.LaurTec.it alla pagina associata al Tutorial ed essere utilizzato per apportare le modifiche che riflettano le vostre specifiche.

```
Imports System.IO        'necessary for Load and Save Settings routines
Imports System.Text      'necessary for Encoding kind
Imports System.IO.Ports  'necessary for Serial Communication
Imports System.Threading 'necessary for Tx and Rx Delays

Public Class frmMain

#Region "Variables"
    ''' <summary>
    ''' List of variables used in this program
    ''' </summary>
    Dim comm As Array
    Dim Even As String
    Dim Odd As String
    Dim None As String
    Dim Mark As String
    Dim Space As String
    Dim Setting As String
#End Region

#Region "User's COM Setting"
    ''' <summary>
    ''' Load port COM settings choosen by user, with all
    ''' values stored in a file named "Settings.txt"
    ''' </summary>
    Private Sub Load_Settings()
        Dim opendlg As New OpenFileDialog()
        If opendlg.ShowDialog = DialogResult.OK Then
            Dim stream As New StreamReader("Settings.txt")
            Dim sLine As String = ""
            Dim arrText As New ArrayList()
            Do
                sLine = stream.ReadLine()
                If Not sLine Is Nothing Then
                    arrText.Add(sLine)
                End If
            Loop Until sLine Is Nothing
            stream.Close()
            cboPort.Text = arrText(0)
            cboBaudrate.Text = arrText(1)
            cboParity.Text = arrText(2)
            cboFormat.Text = arrText(3)
            cboStopbit.Text = arrText(4)
            txtDelayTX.Text = arrText(5)
            txtDelayRX.Text = arrText(6)
        End If
    End Sub

    ''' <summary>
    ''' Save port COM settings choosen by user
```

```
    ''' in a file named "Settings.txt"
    ''' </summary>
    Private Sub Save_Settings()
        Dim Lastvalue(6) As String
        Lastvalue(0) = cboPort.Text
        Lastvalue(1) = cboBaudrate.Text
        Lastvalue(2) = cboParity.Text
        Lastvalue(3) = cboFormat.Text
        Lastvalue(4) = cboStopbit.Text
        Lastvalue(5) = txtDelayTX.Text
        Lastvalue(6) = txtDelayRX.Text
        Dim savedlg As New SaveFileDialog()
        If savedlg.ShowDialog = DialogResult.OK Then
            Dim stream As New StreamWriter("Settings.txt")
            Dim i As Integer
            For i = 0 To Lastvalue.Length - 1
                stream.WriteLine(Lastvalue(i))
            Next
            stream.Close()
        End If
    End Sub
#End Region

#Region "StatusStrip"
    ''' <summary>
    ''' Update Setting label with new values of:
    ''' port COM, Baudrate, Parity, Format and StopBit
    ''' when changing combobox items. Label Status
    ''' is updated with Connect and Disconnect routines
    ''' </summary>
    Private Sub Update_Setting()
        Setting = cboPort.Text & "," & cboBaudrate.Text & "," & cboParity.Text & "," &
cboFormat.Text & "," & cboStopbit.Text
        lblSetting.Text = "RS232 Setting: " & Setting
    End Sub
#End Region

#Region "MenuStrip"
    ''' <summary>
    ''' All MenuStrip Items routines
    ''' </summary>
      Private Sub LoadSettingsToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles LoadSettingsToolStripMenuItem.Click
        Load_Settings()
    End Sub

      Private Sub SaveSettingsToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles SaveSettingsToolStripMenuItem.Click
        Save_Settings()
    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ExitToolStripMenuItem.Click
        If SerialPort1.IsOpen Then
            SerialPort1.DiscardOutBuffer()
            SerialPort1.DiscardInBuffer()
            SerialPort1.Close()
        End If
        Me.Close()
    End Sub

    Private Sub ConnectToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ConnectToolStripMenuItem.Click
        comm_Open()
        If SerialPort1.IsOpen Then
```

```
            txtDelayTX.Enabled = False
            txtDelayTX.BackColor = Me.BackColor
            txtDelayRX.Enabled = False
            txtDelayRX.BackColor = Me.BackColor
            LoadSettingsToolStripMenuItem.Enabled = False
            SaveSettingsToolStripMenuItem.Enabled = False
            ConnectToolStripMenuItem.Enabled = False
            DisconnectToolStripMenuItem.Enabled = True
            Disable_All_Comboboxes()
            txtSend.Enabled = True
            txtSend.Focus()
              Setting = cboPort.Text & "," & cboBaudrate.Text & "," & cboParity.Text &
"," & cboFormat.Text & "," & cboStopbit.Text
            lblSetting.Text = "RS232 Setting: " & Setting
            lblStatus.Text = "   Status: CONNECTED"
        Else
            txtDelayTX.Enabled = True
            txtDelayTX.BackColor = Color.White
            txtDelayRX.Enabled = True
            txtDelayRX.BackColor = Color.White
            LoadSettingsToolStripMenuItem.Enabled = True
            SaveSettingsToolStripMenuItem.Enabled = True
            ConnectToolStripMenuItem.Enabled = True
            DisconnectToolStripMenuItem.Enabled = False
            Enable_All_Comboboxes()
            txtSend.Enabled = False
              Setting = cboPort.Text & "," & cboBaudrate.Text & "," & cboParity.Text &
"," & cboFormat.Text & "," & cboStopbit.Text
            lblSetting.Text = "RS232 Setting: " & Setting
            lblStatus.Text = "   Status: Disconnected"
        End If
    End Sub

    Private Sub DisconnectToolStripMenuItem_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles DisconnectToolStripMenuItem.Click
        If SerialPort1.IsOpen Then
            SerialPort1.DiscardInBuffer()
            SerialPort1.Close()
            txtDelayTX.Enabled = True
            txtDelayTX.BackColor = Color.White
            txtDelayRX.Enabled = True
            txtDelayRX.BackColor = Color.White
            LoadSettingsToolStripMenuItem.Enabled = True
            SaveSettingsToolStripMenuItem.Enabled = True
            ConnectToolStripMenuItem.Enabled = True
            DisconnectToolStripMenuItem.Enabled = False
            Enable_All_Comboboxes()
            txtSend.Enabled = False
            lblStatus.Text = "   Status: Disconnected"
        End If
    End Sub

    Private Sub ClearBufferToolStripMenuItem_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ClearBufferToolStripMenuItem.Click
        rtbDisplay.Clear()
    End Sub
#End Region

#Region "RichTextBox"
    ''' <summary>
    ''' Enable or disable RX buffer clearing
    ''' </summary>
      Private Sub rtbDisplay_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rtbDisplay.TextChanged
        If rtbDisplay.Text <> "" Then
```

```
            ClearBufferToolStripMenuItem.Enabled = True
        Else
            ClearBufferToolStripMenuItem.Enabled = False
        End If
    End Sub
#End Region

#Region "Comboboxes"
    ''' <summary>
    ''' All Comboboxes routines
    ''' </summary>
    Private Sub Enable_All_Comboboxes()
        For Each ctrl As Control In Me.GroupBox3.Controls
            If TypeOf ctrl Is ComboBox Then
                ctrl.Enabled = True
            End If
        Next
    End Sub

    Private Sub Disable_All_Comboboxes()
        For Each ctrl As Control In Me.GroupBox3.Controls
            If TypeOf ctrl Is ComboBox Then
                ctrl.Enabled = False
            End If
        Next
    End Sub

    Private Sub cboPort_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cboPort.SelectedIndexChanged
        Update_Setting()
    End Sub

    Private Sub cboBaudrate_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cboBaudrate.SelectedIndexChanged
        Update_Setting()
    End Sub

    Private Sub cboParity_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cboParity.SelectedIndexChanged
        Update_Setting()
    End Sub

    Private Sub cboFormat_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cboFormat.SelectedIndexChanged
        Update_Setting()
    End Sub

    Private Sub cboStopbit_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cboStopbit.SelectedIndexChanged
        Update_Setting()
    End Sub
#End Region

#Region "Form Main"
    ''' <summary>
    ''' Main form routines
    ''' </summary>
      Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Try
            Set_Port_Default()
        Catch ex As Exception
                MsgBox("No Serial Ports have been found", MsgBoxStyle.Information +
MsgBoxStyle.OkOnly, "Info")
        End Try
```

```
        Update_Setting()
        lblStatus.Text = "   Status: Disconnected"
        LoadSettingsToolStripMenuItem.Enabled = True
        SaveSettingsToolStripMenuItem.Enabled = True
        DisconnectToolStripMenuItem.Enabled = False
        ClearBufferToolStripMenuItem.Enabled = False
        txtSend.Enabled = False
        RadioButton1.Checked = True
    End Sub

       Private Sub frmMain_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing
        If SerialPort1.IsOpen Then
            SerialPort1.DiscardOutBuffer()
            SerialPort1.DiscardInBuffer()
            SerialPort1.Close()
        End If
    End Sub
#End Region

#Region "Buttons"
    ''' <summary>
    ''' Send text, bytes, chars or integer to rs232
    ''' </summary>
     Private Sub btnSend_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSend.Click
        If RadioButton1.Checked = True Then
            Send_Text()
        ElseIf RadioButton2.Checked = True Then
            Send_Bytes(txtSend.Text)
        ElseIf RadioButton3.Checked = True Then
            Convert_Char_to_Int()
        ElseIf RadioButton4.Checked = True Then
            Convert_Int_to_Char(txtSend.Text)
        End If
    End Sub
#End Region

#Region "Radiobuttons"
    ''' <summary>
    ''' set the TX buffer max value of digits and set the focus on it
    ''' </summary>
     Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton1.CheckedChanged
        txtSend.MaxLength = 32767
        txtSend.Focus()
    End Sub

     Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton2.CheckedChanged
        txtSend.MaxLength = 3 'just three numbers
        txtSend.Focus()
    End Sub

     Private Sub RadioButton3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton3.CheckedChanged
        txtSend.MaxLength = 1 'just one char
        txtSend.Focus()
    End Sub

     Private Sub RadioButton4_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadioButton4.CheckedChanged
        txtSend.MaxLength = 3 'just three numbers
        txtSend.Focus()
    End Sub
```

```
#End Region

#Region "Set Port Default"
    ''' <summary>
    ''' Set port COM with default values (COM1,19200,None,8,1)
    ''' </summary>
    Private Sub Set_Port_Default()
        'Get all com ports available
        comm = IO.Ports.SerialPort.GetPortNames()
        For i = 0 To UBound(comm)
            cboPort.Items.Add(comm(i))
        Next
        'Fill cboBaudrate combobox with all values listed below
        cboBaudrate.Items.Add(110)
        cboBaudrate.Items.Add(300)
        cboBaudrate.Items.Add(600)
        cboBaudrate.Items.Add(1200)
        cboBaudrate.Items.Add(2400)
        cboBaudrate.Items.Add(4800)
        cboBaudrate.Items.Add(9600)
        cboBaudrate.Items.Add(14400)
        cboBaudrate.Items.Add(19200)
        cboBaudrate.Items.Add(28800)
        cboBaudrate.Items.Add(38400)
        cboBaudrate.Items.Add(56000)
        cboBaudrate.Items.Add(57600)
        cboBaudrate.Items.Add(115200)
        'Fill cboParity combobox with all values listed below
        cboParity.Items.Add("Even")
        cboParity.Items.Add("Odd")
        cboParity.Items.Add("None")
        cboParity.Items.Add("Mark")
        cboParity.Items.Add("Space")
        'Fill cboFormat combobox with all values listed below
        cboFormat.Items.Add(5)
        cboFormat.Items.Add(6)
        cboFormat.Items.Add(7)
        cboFormat.Items.Add(8)
        'Fill cboStopbit combobox with all values listed below
        cboStopbit.Items.Add(1)
        cboStopbit.Items.Add(2)
        'Set cboPort to the COM1 port (Sorted = True)
        cboPort.Sorted = True
        cboPort.Text = cboPort.Items.Item(0)
        'Set cboBaudrate to 19200
        cboBaudrate.Text = cboBaudrate.Items.Item(8)
        'Sets cboParity to None
        cboParity.Text = cboParity.Items.Item(2)
        'Set cboFormat to 8
        cboFormat.Text = cboFormat.Items.Item(3)
        'Set cboStopbit to 1
        cboStopbit.Text = cboStopbit.Items.Item(0)
        txtDelayTX.Text = 100
        txtDelayRX.Text = 10
    End Sub
#End Region

#Region "Open port"
    ''' <summary>
    ''' Function to open port communication
    ''' </summary>
    Private Function comm_Open() As Boolean
        Try
            SerialPort1.PortName = cboPort.Text
```

```
            SerialPort1.BaudRate = cboBaudrate.Text
            SerialPort1.Parity = cboParity.SelectedValue
            SerialPort1.DataBits = cboFormat.Text
            SerialPort1.StopBits = cboStopbit.Text
            SerialPort1.ReadTimeout = ((Val(txtDelayRX.Text)) * 2)
            'set Encoding to UTF8
            SerialPort1.Encoding = System.Text.Encoding.UTF8
            If SerialPort1.IsOpen = True Then
                SerialPort1.Close()
            End If
            SerialPort1.Open()
            Return True
        Catch ex As Exception
              MsgBox(cboPort.Text & " " & "Port access denied", MsgBoxStyle.Critical +
MsgBoxStyle.OkOnly, "Error")
            Return False
        End Try
    End Function
#End Region

#Region "Send Text"
    ''' <summary>
    ''' Send data as text to rs232
    ''' </summary>
    Private Sub Send_Text()
        Try
            SerialPort1.Write(txtSend.Text & vbCrLf)
            With rtbDisplay
                .SelectionColor = Color.Black
                .AppendText(txtSend.Text & vbCrLf)
                .ScrollToCaret()
            End With
            txtSend.Text = String.Empty
            txtSend.Focus()
        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
        End Try
    End Sub
#End Region

#Region "Send Bytes"
    ''' <summary>
    ''' Send data as byte to rs232
    ''' </summary>
    Public Sub Send_Bytes(ByVal message As String)
        If IsNumeric(txtSend.Text) = False Then
            MsgBox("Attention: you cannot enter chars, but only numbers less than 256",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
            txtSend.Text = String.Empty
            txtSend.Focus()
        End If
        If IsNumeric(txtSend.Text) = True And Val(message) > 255 Then
                    MsgBox("Attention: the number you entered is more than 255",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
            txtSend.Text = String.Empty
            txtSend.Focus()
        Else
            Dim byte_to_send(1) As Byte
            byte_to_send(1) = Val(message)
            Try
                SerialPort1.Write(byte_to_send, 1, 1)
                With rtbDisplay
                    .SelectionColor = Color.Black
                    .AppendText(txtSend.Text & vbCrLf)
                    .ScrollToCaret()
```

```
                End With
                txtSend.Text = String.Empty
                txtSend.Focus()
            Catch ex As Exception
                    MsgBox(ex.Message, MsgBoxStyle.Information + MsgBoxStyle.OkOnly,
"Info")
            End Try
        End If
    End Sub
#End Region

#Region "Convert Char to Int"
    ''' <summary>
    ''' Send one char converted to int to rs232
    ''' </summary>
    Private Sub Convert_Char_to_Int()
        Try
                 If IsNumeric(txtSend.Text) = True And (txtSend.Text) < Chr(33) Or
(txtSend.Text) > Chr(126) Then
                 MsgBox("Attention: you can enter only chars under the UTF8 encoding",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
                txtSend.Text = String.Empty
                txtSend.Focus()
            Else
                SerialPort1.Write(txtSend.Text)
                With rtbDisplay
                    .SelectionColor = Color.Black
                    .AppendText(txtSend.Text & vbCrLf)
                    .ScrollToCaret()
                End With
                txtSend.Text = String.Empty
                txtSend.Focus()
            End If
        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
        End Try
    End Sub
#End Region

#Region "Convert Int to Char"
    ''' <summary>
    ''' Send int converted to char to rs232
    ''' </summary>
    Public Sub Convert_Int_to_Char(ByVal message As String)
        Try
            If IsNumeric(txtSend.Text) = False Then
                txtSend.Text = String.Empty
                txtSend.Focus()
            End If
            If IsNumeric(txtSend.Text) = True And (txtSend.Text) < 33 Or (txtSend.Text)
> 126 Or (txtSend.Text) > 256 Then
                 MsgBox("Attention: you can only enter integers between 33 and 126",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
                txtSend.Text = String.Empty
                txtSend.Focus()
            Else
                Dim byte_to_send(1) As Byte
                byte_to_send(1) = Val(message)
                SerialPort1.Write(byte_to_send, 1, 1)
                With rtbDisplay
                    .SelectionColor = Color.Black
                    .AppendText(txtSend.Text & vbCrLf)
                    .ScrollToCaret()
                End With
                txtSend.Text = String.Empty
```

```
                txtSend.Focus()
            End If
        Catch ex As Exception
                    MsgBox("Attention: you cannot enter chars, but only integers",
MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Info")
        End Try
    End Sub
#End Region

#Region "UpdateRTB"
    ''' <summary>
    ''' Delegate and subroutine to update the RichTextBox control
    ''' </summary>
    Public Delegate Sub myDelegate()
    Public Sub updateRTB()
        Try
            With rtbDisplay
                If RadioButton1.Checked = True Then
                    .SelectionColor = Color.Red
                    .AppendText(SerialPort1.ReadExisting)
                    Thread.Sleep(Val(txtDelayRX.Text))
                    .ScrollToCaret()
                ElseIf RadioButton2.Checked = True Then
                    .SelectionColor = Color.Green
                    .AppendText(SerialPort1.ReadByte)
                    Thread.Sleep(Val(txtDelayRX.Text))
                    .AppendText(vbCr)
                    txtSend.Text = String.Empty
                    txtSend.Focus()
                    .ScrollToCaret()
                ElseIf RadioButton3.Checked = True Then
                    .SelectionColor = Color.Blue
                    .AppendText(SerialPort1.ReadByte & vbCrLf)
                    Thread.Sleep(Val(txtDelayRX.Text))
                    txtSend.Text = String.Empty
                    txtSend.Focus()
                    .ScrollToCaret()
                ElseIf RadioButton4.Checked = True Then
                    .SelectionColor = Color.Maroon
                    .AppendText(SerialPort1.ReadExisting)
                    Thread.Sleep(Val(txtDelayRX.Text))
                    .AppendText(vbCr)
                    txtSend.Text = String.Empty
                    txtSend.Focus()
                    .ScrollToCaret()
                End If
            End With
        Catch ex As Exception
                MsgBox("Data reading error", MsgBoxStyle.Critical + MsgBoxStyle.OkOnly,
"Error")
            SerialPort1.DiscardInBuffer()
        End Try
        Exit Sub
    End Sub
#End Region

End Class
```

Debug e Compilazione

Se tutto è stato fatto correttamente, facendo un click su Debug e poi su Avvia Debug, dovremo vedere avviarsi l'applicazione senza nessun messaggio di errore. Se il collaudo, descritto più avanti, dovesse avere esito positivo si può procedere alla fase di compilazione facendo un click sul Compila e poi su Compila SerialPort. Il file esecutivo dell'applicazione verrà creato nella directory bin\release.

Alcune note sul programma

La codifica utilizzata è UTF8, per cui verranno riconosciuti solo i caratteri compresi tra 0x21 e 0x7F (ovvero caratteri compresi tra int 33 e int 126).

Nel caso Write Bytes, se per errore viene inserito un carattere, si avrà un valore di ritorno, ovvero risultato restituito, pari a 0 (zero)

Nel caso Convert Char to Int, anche i numeri dallo 0 al 9 verranno trattati come caratteri, restituendo così il loro rispettivo valore intero

Tutti i dati in trasmissione verranno inviati a RX Buffer con colore Nero.

I dati in ricezione verranno inviati a RX Buffer con colori diversi, secondo il dato ricevuto:

Rosso = Text
Verde = Bytes
Blu = Chars to Int
Marrone = Int to Char

La compilazione è stata effettuata utilizzando il Framework 3.5, in modo da non richiedere l'installazione del Framework .NET, presente di Default nelle ultime versioni di ogni sistema operativo Windows.

Collaudo dell'Applicazione

Il collaudo dell'interfaccia, senza la scheda Freedom, può essere fatto utilizzando un cavo Seriale DB9 alla porta seriale del PC, cortocircuitando i pin 2 e 3 (RX e TX) con un ponticello. Una volta avviata l'applicazione ed abilitata la connessione tramite "pressione" del pulsante Connect, sarà possibile inviare i vari tipi di dato selezionando il rispettivo radiobutton. Un esempio dell'applicazione in esecuzione è riportato in Figura 7.

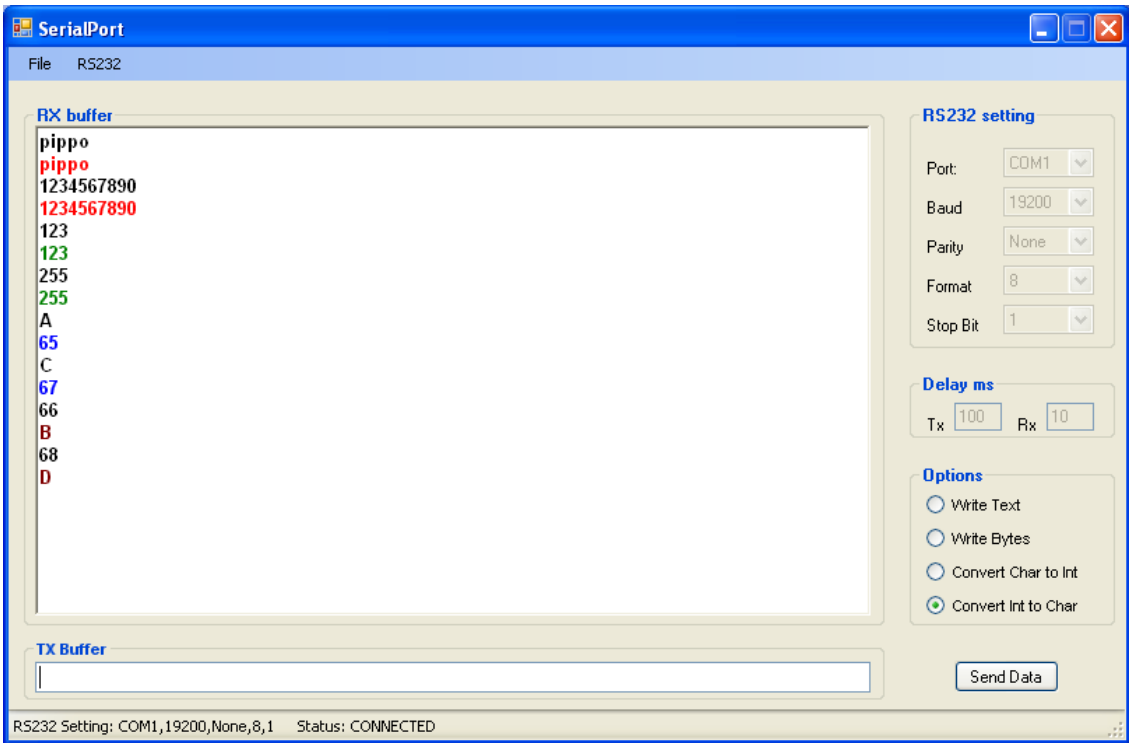

Figura 7: *Esempio dell'applicazione in esecuzione.*

Se invece volessimo collaudare l'applicazione utilizzando la scheda Freedom, senza scrivere un firmware ad essa dedicato, possiamo utilizzare il file compilato del progetto "Controllo otto servi", presente nella sezione Progetti degli utenti del sito www.LaurTec.it. Dopo avere programmato il PIC18F4550 con il file "controllo_otto_servi.X.production.hex", colleghiamo un servomotore al primo pin (RD0) della PORTD del microcontrollore, corrispondente al servo n° 1, quindi con indice = 1.

Nota

Se il servo viene alimentato separatamente, la sua massa e quella della scheda devono essere messe in comune.

Infine colleghiamo, tramite cavo seriale DB9 la porta seriale del PC al connettore femmina DB9 montato a bordo della scheda. A questo punto avviamo l'interfaccia, ci connettiamo, selezioniamo l'opzione Write Bytes ed inviamo, in successione, prima l'indice del servo e poi i ms: se tutto è a posto, si vedrà il servo spostarsi nella posizione voluta, avendo così la conferma che esiste comunicazione tra Software e Hardware, come riportato in Figura 8.

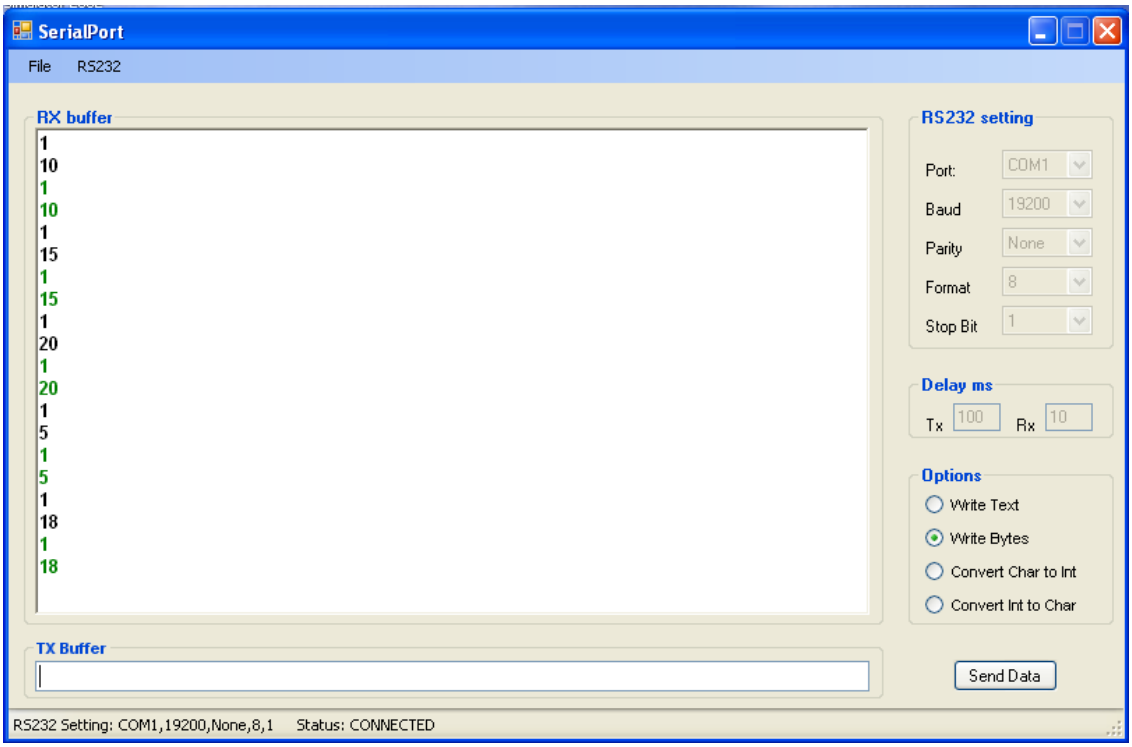

Figura 8: *Esempio di collaudo facendo uso della scheda Freedom I.*

Conclusione

Ora che siamo giunti alla fine di questo Tutorial, vorrei ricordare che questo, in realtà, è solo un inizio: nessuno, infatti, ci vieta di usare altri controlli o di aggiungerne altri, oppure di modificare le routine a nostro piacimento o di riscriverle di sana pianta. Il tutto, credo, si può riassumere con una sola parola: sperimentare, senza avere la paura di sbagliare. Nel frattempo, resto a vostra disposizione per eventuali domande, chiarimenti e, in modo particolare, segnalazioni di bug o di errori nel programma.

Bibliografia

[1] www.LaurTec.it : sito ufficiale del Tutorial UT0001 dove poter scaricare ogni aggiornamento e l'applicazione di esempio.
[2] www.microchip.com : sito dove scaricare i datasheet del PIC18F4550.

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
25.04.14	1.0	Marcello Pinna	Mauro Laurenti	Versione originale.