

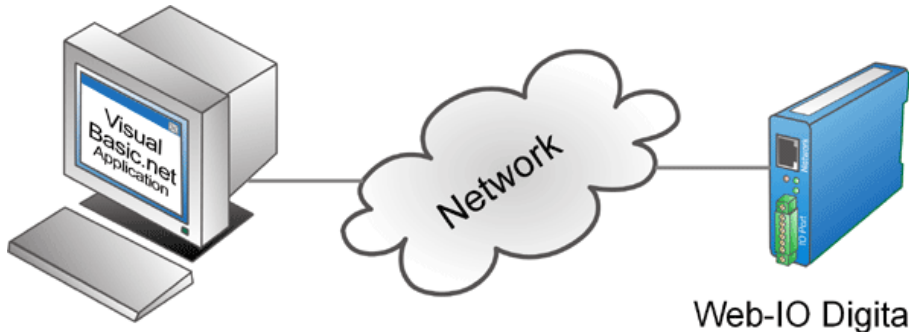
Applicazione relativa al Web-IO digitale:

# Web-IO digitale con VB.Net controllo e monitoraggio

Panoramica del prodotto

Ulteriori applicazioni

Come successore di MS Visual Basic nel frattempo si è imposto VB.net. Visual Basic.net offre tutto ciò che è necessario per la programmazione di applicazioni TCP/IP. In tal modo Visual Basic.net è uno strumento ausiliario prediletto per la creazione di applicazioni che comunicano con il **Web-IO digitale**. Ulteriori driver o DLL non saranno necessari.



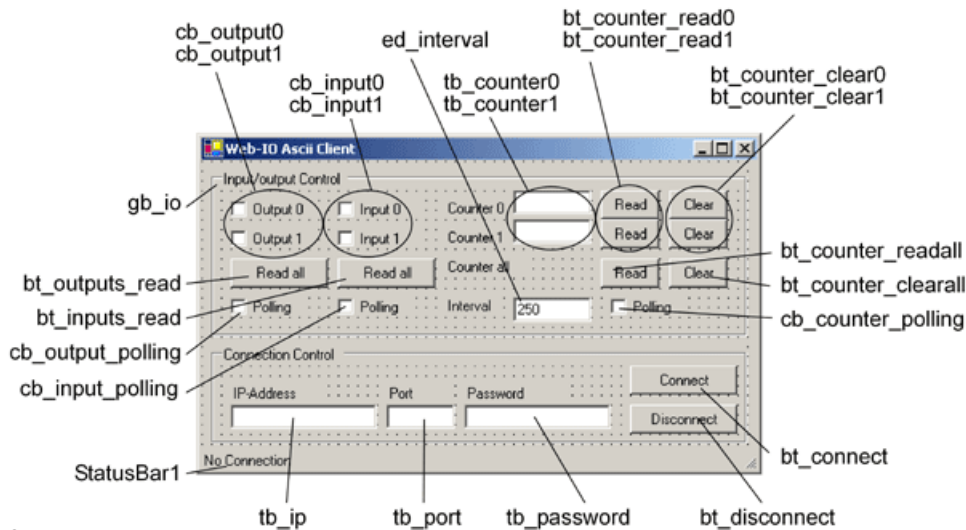
Con il seguente esempio di programma potete riprodurre il vostro Web-IO digitale con i suoi input e output in un'applicazione Windows. Inoltre potete collegare gli output del Web-IO.

## Preparativi

Avete già alimentato con corrente

- il vostro Web-IO digitale
- collegate gli ingressi e le uscite
- effettuato il collegamento alla vostra rete
- e assegnato un indirizzo IP: con WuTility è facile!

## 1. Collocazione dei diversi elementi di comando e oggetti di visualizzazione nel modulo VB.net



Oltre agli oggetti qui mostrati il programma necessita di un ulteriore timer per il polling (timer\_polling).

Nella denominazione dei singoli oggetti è utile utilizzare nomi che ne riprendono il significato. In questo esempio la prima parte del nome descrive il tipo di oggetto e la seconda parte la funzione.

## 1. Avvio del programma

Mentre la costruzione delle interfacce utente grafiche in VB.net è ugualmente facile da realizzare come con VB5 o VB6, la comunicazione di rete sotto VB.net risulta leggermente più difficile rispetto alla versione precedente. Ciò dipende innanzitutto dal fatto che Microsoft con VB.net non mette più a disposizione l'elemento di controllo Winsock. Per l'accesso alla rete mediante la proposizione *Import* deve invece essere istanziato nell'intestazione del testo sorgente il namespace per le classi socket utilizzate.

Inoltre deve essere creato il socket, su cui deve svolgersi la comunicazione, e deve essere definito un buffer per i dati di ingresso.

```
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Windows.Forms
Public Class Form1
Inherits System.Windows.Forms.Form
Dim TCP_client As Socket
Dim receivebuffer(511) As Byte
```

## 2. Controllo del collegamento

Inizializzazione del collegamento

Immettendo l'indirizzo IP del Web-IO nel campo di testo `ed_ip` e facendo clic sul pulsante `bt_connect` viene avviata la creazione del collegamento

```
Private Sub bt_connect_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles bt_connect.Click
    Dim webioep As New IPEndPoint(IPAddress.Parse(tb_ip.Text),
    Val(tb_port.Text))
    If tb_ip.Text <> "" And tb_port.Text <> "" Then
        TCP_client = New Socket(AddressFamily.InterNetwork, SocketType.Stream, _
        ProtocolType.Tcp)
        bt_connect.Enabled = False
        Try
            TCP_client.BeginConnect(webioep, New AsyncCallback(AddressOf callback_connect), _
            TCP_client)
        Catch ex As Exception
            closeconnections()
            MessageBox.Show(ex.Message, "Socket Error", MessageBoxButtons.OK, _
            MessageBoxIcon.Error)
        End Try
    End If
End Sub
```

Creazione del collegamento

Per lo svolgimento della gestione TCP/IP viene innanzitutto definito un `IPEndPoint` dall'indirizzo `Ip` e dalla porta `TCP` e con ciò viene inizializzato il socket `TCP_client`. Durante la richiesta di collegamento viene creato un rimando a una procedura di callback.

Collegamento realizzato

Non appena il Web-IO accetta il collegamento, viene eseguita la procedura di callback. Nella riga di stato viene visualizzata la realizzazione del collegamento, gli elementi di comando vengono abilitati per l'utilizzo e il pulsante `Disconnect` risulta utilizzabile. Il pulsante `Connect` viene bloccato per l'uso. Inoltre viene creato un rimando a una routine di callback per la ricezione dei dati.

```
Private Sub callback_connect(ByVal ar As IAsyncResult)
    bt_connect.Enabled = False
    gb_io.Enabled = True
    bt_disconnect.Enabled = True
    StatusBar1.Text = "Connected to " + tb_ip.Text + " : " + tb_port.Text
    timer_polling.Enabled = True
    Try
        TCP_client.EndConnect(ar)
        TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, New AsyncCallback(AddressOf
        callback_readdata), TCP_client)
    Catch ex As Exception
        closeconnections()
        StatusBar1.Text = "error on connecting"
    End Try
End Sub
```

Disinserzione del collegamento

Il collegamento rimane fino a quando non viene terminato dall'utente facendo clic sul pulsante `Disconnect` oppure il Web-IO termina il collegamento.

```
Private Sub bt_disconnect_Click(ByVal sender
    As System.Object, _
    ByVal e As System.EventArgs) Handles bt_disconnect.Click
    bt_disconnect.Enabled = False
    closeconnections()
End Sub
```

In questo caso viene richiamata una corrispondente procedura.

```
Private Sub closeconnections()
    Dim ar As IAsyncResult
    timer_polling.Enabled = False
    Try
        TCP_client.EndReceive(ar)
    Catch ex As Exception
    End Try
    Try
        TCP_client.Shutdown(SocketShutdown.Both)
    Catch ex As Exception
    End Try
    Try
        TCP_client.Close()
    Catch ex As Exception
    End Try
    bt_connect.Enabled = True
    gb_io.Enabled = False
    bt_disconnect.Enabled = False
    StatusBar1.Text = "no connection"
End Sub
```

Errore di collegamento

Tutte le azioni che riguardano la comunicazione TCP/IP vengono eseguite nell'ambito dell'istruzione `Try`. Se compaiono errori, viene richiamata anche la procedura `CloseConnection`.

### 3. Utilizzo e comunicazione della parte client

Non appena viene realizzato un collegamento con il Web-IO, l'utente può inviare comandi al Web-IO utilizzando i corrispondenti

elementi del programma.

```
Private Sub sendcommand(ByVal sendstring As String)
Dim senddata As Byte() = System.Text.Encoding.ASCII.GetBytes(sendstring)
Try
    TCP_client.Send(senddata)
Catch ex As Exception
    closeconnections()
End Try
End Sub
```

Impostazione degli output

L'impostazione degli output è resa possibile all'utente da due caselle di spunta *cb\_outputx*. Il programma utilizza a tale scopo l'evento MouseUP di questo oggetto. Se viene registrato un MouseUp, ossia un abbandono della casella di spunta degli output, il programma esegue la corrispondente procedura e inoltra al Web-IO, in base all'eventuale impostazione della casella di spunta, il comando adatto.

```
Private Sub cb_output0_MouseUp(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles cb_output0.MouseUp
If cb_output0.Checked Then
    sendcommand("GET /outputaccess0?PW="+ tb_password.Text + "&State=ON&")
Else
    sendcommand("GET /outputaccess0?PW="+ tb_password.Text + "&State=OFF&")
End If
End Sub

Private Sub cb_output1_MouseUp(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles cb_output1.MouseUp
If cb_output1.Checked Then
    sendcommand("GET /outputaccess1?PW="+ tb_password.Text + "&State=ON&")
Else
    sendcommand("GET /outputaccess1?PW="+ tb_password.Text + "&State=OFF&")
End If
End Sub
```

Interrogazione dello stato degli output/input

L'utente può richiedere lo stato degli output e degli input facendo clic sul relativo pulsante.

```
Private Sub bt_outputs_read_Click(ByVal
sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_outputs_read.Click
    sendcommand("GET /output?PW=" + tb_password.Text + "&")
End Sub

Private Sub bt_inputs_read_Click(ByVal
sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_inputs_read.Click
    sendcommand("GET /input?PW=" + tb_password.Text + "&")
End Sub
```

Interrogazione/cancellazione dei counter

È possibile interrogare o cancellare anche gli stati dei counter degli input.

```
Private Sub bt_counter_read0_Click(ByVal sender As System.Object,
_
ByVal e As System.EventArgs) Handles bt_counter_read0.Click
    sendcommand("GET /counter0?PW=" + tb_password.Text + "&")
End Sub

Private Sub bt_counter_read1_Click(ByVal sender
As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_read0.Click
    sendcommand("GET /counter1?PW=" + tb_password.Text + "&")
End Sub

Private Sub bt_counter_clear0_Click(ByVal sender As System.Object,
_
ByVal e As System.EventArgs) Handles bt_counter_clear0.Click
    sendcommand("GET /counterclear0?PW=" + tb_password.Text + "&")
End Sub

Private Sub bt_counter_clear1_Click(ByVal
sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_clear0.Click
    sendcommand("GET /counterclear1?PW=" + tb_password.Text + "&")
End Sub
```

Naturalmente è possibile leggere o cancellare contemporaneamente anche tutti i counter.

```
Private Sub bt_counter_readall_Click(ByVal
sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_readall.Click
    sendcommand("GET /counter?PW=" + tb_password.Text + "&")
End Sub
```

```

Private Sub bt_counter_clearall_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_clearall.Click
    sendcommand("GET /counterclear?PW=" + tb_password.Text + "&")
End Sub

```

### 3. Ricezione dei dati dal Web-IO

Analisi e visualizzazione dei dati ricevuti

- Tutti i comandi e le richieste al Web-IO vengono confermati con una stringa di risposta. Le risposte hanno una struttura specifica in base al tipo.
- Per gli output: output;<valore binario dello stato degli output in formato esadecimale>
- Per gli input: input;<valore binario dello stato degli input in formato esadecimale>
- Per i counter: counter;<stato del conteggio decimale>
- oppure counter;<stato del conteggio decimale 0 >; <stato del conteggio decimale 0 >; ... se tutti i counter devono essere letti in un'unica volta.
- Tutte le stringhe di risposta terminano con 0 byte.
- Alla ricezione dei dati viene richiamata la corrispondente procedura di callback

```

Private Sub callback_readdata(ByVal ar As IAsyncResult)
If TCP_client.Connected Then
Dim bytesread As Integer
Try
bytesread = TCP_client.EndReceive(ar)
Catch ex As Exception
End Try
If bytesread = 0 Then
closeconnections()
Else
Dim receivestring As String
receivestring = System.Text.Encoding.ASCII.GetString(receivebuffer, 0, _
receivebuffer.Length)
receivebuffer.Clear(receivebuffer, 0, 512)
Try
TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, New AsyncCallback(AddressOf
callback_readdata), TCP_client)
Catch ex As Exception
closeconnections()
End Try
Select Case Mid(receivestring, 1, 1)
Case "i"
If (Val(Mid(receivestring, 7, 1)) And 1) = 1 Then
cb_input0.Checked = True
Else
cb_input0.Checked = False
End If
If (Val(Mid(receivestring, 7, 1)) And 2) = 2 Then
cb_input1.Checked = True
Else
cb_input1.Checked = False
End If
Case "o"
If (Val(Mid(receivestring, 8, 1)) And 1) = 1 Then
cb_output0.Checked = True
Else
cb_output0.Checked = False
End If
If (Val(Mid(receivestring, 8, 1)) And 2) = 2 Then
cb_output1.Checked = True
Else
cb_output1.Checked = False
End If
Case "c"
Dim tabpos
If Mid(receivestring, 8, 1) = "0" Then _
tb_counter0.Text = Mid(receivestring, 10)
If Mid(receivestring, 8, 1) = "1" Then _
tb_counter1.Text = Mid(receivestring, 10)
If Mid(receivestring, 8, 1) = ":" Then
tabpos = InStr(9, receivestring, ";")
tb_counter0.Text = Mid(receivestring, 9, tabpos - 9)
tb_counter1.Text = Mid(receivestring, tabpos + 1, _
Len(receivestring) - tabpos - 1)
End If
End Select
End If
End If
End Sub

```

La procedura di ricezione controlla sulla base del primo carattere dei dati ricevuti se si tratta di messaggi degli input, degli output o dei counter. In base a ciò viene ad es. stabilito quale stato ha quale output. Nei counter è possibile sia interrogare i valori dei singoli contatori che leggere tutti i counter contemporaneamente. Gli stati dei singoli contatori vengono visualizzati in una stringa in decimali separati mediante punto e virgola.

### 4. Polling

Interrogazione ciclica di determinati valori

Per permettere anche un aggiornamento automatico della visualizzazione, viene utilizzato un timer.

In base alle caselle di spunta per il polling degli output, degli input e dei counter le corrispondenti informazioni vengono interrogate dal Web-IO nell'intervallo impostato.

```
Private Sub timer_polling_Elapsed(ByVal sender
As System.Object, _
ByVal e As System.Timers.ElapsedEventArgs) Handles timer_polling.Elapsed
If (cb_input_polling.Checked And TCP_client.Connected) Then
sendcommand("GET /input?PW=" + tb_password.Text + "&")
End If
If (cb_output_polling.Checked And TCP_client.Connected) Then
sendcommand("GET /output?PW=" + tb_password.Text + "&")
End If
If (cb_output_polling.Checked And TCP_client.Connected) Then
sendcommand("GET /output?PW=" + tb_password.Text + "&")
End If
If (cb_counter_polling.Checked And TCP_client.Connected) Then
sendcommand("GET /counter?PW=" + tb_password.Text + "&")
End If
End Sub
```

L'intervallo desiderato può essere immesso nel corrispondente campo di testo. In caso di una modifica l'intervallo del timer viene adattato automaticamente.

```
Private Sub tb_interval_TextChanged(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles tb_interval.TextChanged
timer_polling.Interval = Val(tb_interval.Text)
End Sub
```

Il [↓ programma esempio](#) supporta tutte le comuni funzioni del Web-IO nella modalità stringa di comando, ottimizzata per **iWeb-IO 2x input digitale, 2x output digitale**. Per gli altri modelli di Web-IO devono eventualmente essere eseguiti adattamenti al programma. Ulteriori esempi di programma per la programmazione socket sono riportati nelle [pagine dei tool](#) per il Web-IO. Una descrizione dettagliata sull'interfaccia socket dei modelli Web-IO digitali è riportata nel [manuale di riferimento](#).



Saremo lieti di fornirvi una consulenza personalizzata!

Wiesemann & Theis  
GmbH  
Porschestr. 12  
42279 Wuppertal  
Tel.: +49 202/2680-110 (Lun-Ven. 8-17)  
Fax: +49 202/2680-265  
info@wut.de

© Wiesemann & Theis GmbH, con riserva di errori e modifiche: poiché possono verificarsi errori, nessuna nostra informazione deve essere utilizzata senza essere stata verificata. Vi preghiamo di comunicarci tutti gli errori o gli equivoci che avete rilevato in modo tale che possiamo riconoscerli ed eliminarli quanto prima.

[Protezione dei dati](#)