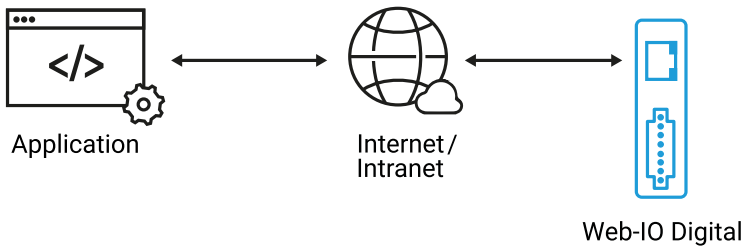
Tutorial al Web-IO digitale:

Web-IO digitale controllo e monitoraggio con Delphi

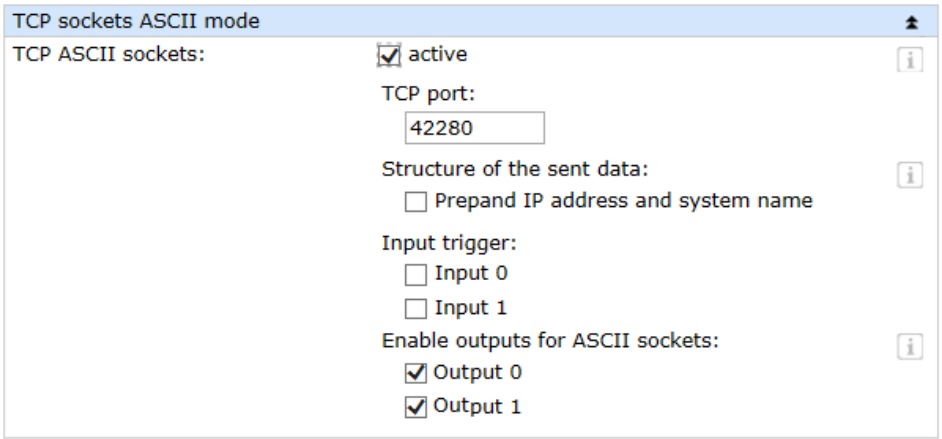
Come linguaggio di alto livello di facile apprendimento, Delphi offre tutto ciò che è necessario per la programmazione di applicazioni TCP/IP. In tal modo Delphi è anche uno strumento ausiliario prediletto per la creazione di applicazioni che comunicano con il Web-IO digitale. Ulteriori driver o DLL non saranno necessari.



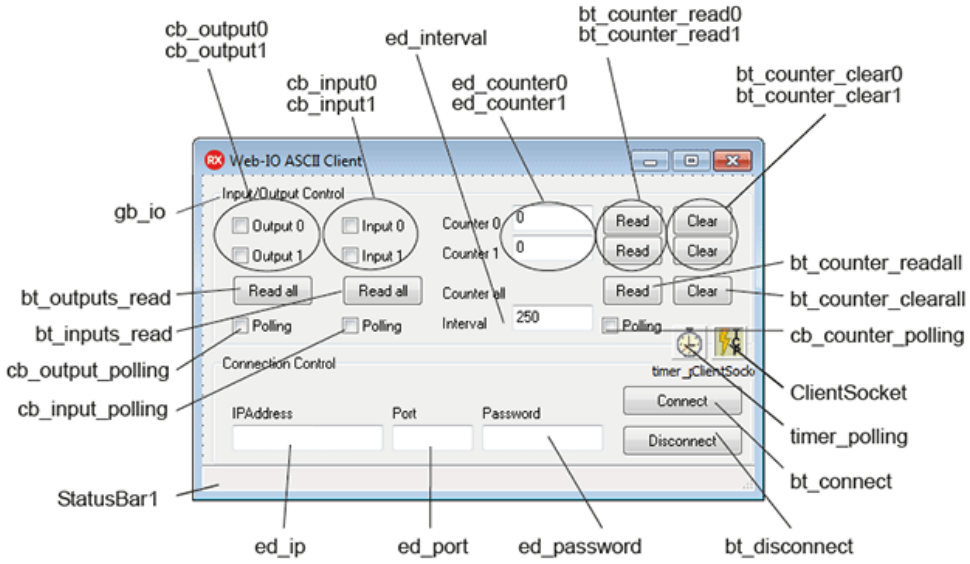
Con il seguente esempio di programma potete riprodurre il vostro Web-IO digitale con i suoi input e output in un'applicazione Windows. Inoltre potete collegare gli output del Web-IO.

Preparativi

- Mettere sotto tensione il Web-IO e collegare gli IO
- Collegare il Web-IO alla rete
- Assegnazione di indirizzi IP
- Nel Web-IO nell'area *Vie di comunicazione >> API socket* Attivare i *socket ASCII TCP* e *autorizzare gli output per l'attivazione*



Collocazione dei diversi elementi di comando e oggetti di visualizzazione nel modulo Delphi



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 dell'oggetto e la seconda parte la funzione.

Per l'attuazione della comunicazione TCP viene usata la componente *TIdTCPClient* di Indy che appartiene ai componenti Delphi.

Variabili e procedure globali

Innanzitutto è necessario dichiarare alcune variabili globali.

```
var
  .......
  ClientSocketThread : TIdThreadComponent;
  Receivestring : string;
```

Importante è la componente thread *ClientSocketThread*, per la quale deve essere creata una rispettiva procedura. Più tardi questo Thread eseguirà la ricezione asincrona dei dati di rete in entrata.

```
private
  { Private-Deklarationen }
public
  procedure ClientSocketThreadRun(Sender: TIdThreadComponent);
end;
```

La rispettiva procedura deve essere creata nell'area *implementation* (vedere sotto).

Avvio del programma

Inizializzazione degli elementi di comando
Il gruppo con gli elementi di comando per il Web-IO viene innanzitutto bloccato per l'uso. Nella riga di stato viene visualizzato che non sussiste ancora alcun collegamento.

```
procedure Twebio_ascii_client.FormCreate(Sender: TObject);
begin
  ClientSocketThread := TIdThreadComponent.Create();
  ClientSocketThread.onRun := ClientSocketThreadRun;
  StatusBar1.SimpleText := 'No Connection';
  bt_disconnect.Enabled := False;
  gb_io.Enabled := False;
end;
```

Controllo del collegamento

Inizializzazione del collegamento
Immettendo l'indirizzo IP del Web-IO nel campo di testo *ed_ip* e della porta TCP nel campo di testo *ed_port* e facendo clic sul pulsante *bt_connect* viene avviata la creazione del collegamento

```
procedure Twebio_ascii_client.bt_connectClick(Sender: TObject);
begin
  if ed_ip.Text <> '' then
  begin
    ClientSocket.Host := ed_ip.Text;
    ClientSocket.Port := strtoint(ed_port.Text);
    ClientSocket.Connect;
  end;
end;
```

Collegamento realizzato
Non appena il Web-IO accetta il collegamento, l'elemento di controllo ClientSocket esegue la corrispondente procedura. 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.

```
procedure Twebio_ascii_client.ClientSocketConnected(Sender: TObject);
begin
  ClientSocketThread.Active := true;
  StatusBar1.SimpleText := 'Connected to ' + ed_ip.Text;
  bt_connect.Enabled := False;
  bt_disconnect.Enabled := True;
  gb_io.Enabled := True;
end;
```

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.

```
procedure Twebio_ascii_client.bt_disconnectClick(Sender: TObject);
begin
  ClientSocket.Disconnect;
end;
```

Anche in questo caso l'elemento di controllo ClientSocket richiama la corrispondente procedura.

```
procedure Twebio_ascii_client.ClientSocketDisconnected(Sender: TObject);
begin
  ClientSocketThread.Active := false;
  ClientSocket.Disconnect;
  StatusBar1.SimpleText := 'No Connection';
  bt_connect.Enabled := True;
  bt_disconnect.Enabled := False;
  gb_io.Enabled := False;
end;
```

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.

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.

```
procedure Twebio_ascii_client.cb_outputMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if sender = cb_output0 then
    if cb_output0.Checked then
      ClientSocket.IOHandler.
          Write('GET /outputaccess0?PW=' + ed_password.Text + '&State=ON&')
    else
      ClientSocket.IOHandler.
          Write('GET /outputaccess0?PW=' + ed_password.Text + '&State=OFF&')
  else
    if cb_output1.Checked then
      ClientSocket.IOHandler.
          Write('GET /outputaccess1?PW=' + ed_password.Text + '&State=ON&')
    else
      ClientSocket.IOHandler.
          Write('GET /outputaccess1?PW=' + ed_password.Text + '&State=OFF&');
end;;
```

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

```
procedure Twebio_ascii_client.bt_outputs_readClick(Sender: TObject);
begin
  ClientSocket.IOHandler.Write('GET /output?PW=' + ed_password.Text + '&');
end;


procedure Twebio_ascii_client.bt_inputs_readClick(Sender: TObject);
begin
  ClientSocket.IOHandler.Write('GET /input?PW=' + ed_password.Text + '&');
end;
```

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

```
procedure Twebio_ascii_client.bt_counter_readClick(Sender: TObject);
begin
  if sender = bt_counter_read0 then
    ClientSocket.IOHandler.Write('GET /counter0?PW=' + ed_password.Text + '&')
  else
    ClientSocket.IOHandler.Write('GET /counter1?PW=' + ed_password.Text + '&');
end;


procedure Twebio_ascii_client.bt_counter_clearClick(Sender: TObject);
begin
  if sender = bt_counter_clear0 then
    ClientSocket.IOHandler.Write('GET /counterclear0?PW=' + ed_password.Text + '&')
  else
    ClientSocket.IOHandler.Write('GET /counterclear1?PW=' + ed_password.Text + '&');
end;
```

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

```
procedure Twebio_ascii_client.bt_counter_readallClick(Sender: TObject);
begin
  ClientSocket.IOHandler.Write('GET /counter?PW=' + ed_password.Text + '&')
end;


procedure Twebio_ascii_client.bt_counter_clearallClick(Sender: TObject);
begin
  ClientSocket.IOHandler.Write('GET /counterclear?PW=' + ed_password.Text + '&')
end;
```

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. È responsabile della ricezione e della valutazione delle risposte il thread menzionato all'inizio *ClientSocketThreadRun*.

Formato delle risposte
Le risposte hanno una struttura specifica in base al tipo.

- Per gli output: output;<Binärwert des Outputstatus im hexadezimalen Format>
- Per gli input: input;<Binärwert des Outputstatus im hexadezimalen Format>
- Per i counter: counterx;<dezimaler Zählerstand>
- o counter;<dezimaler Zählerstand 0 >; <dezimaler Zählerstand 0 >; ..... ,
  se tutti i counter devono essere letti contemporaneamente.

Tutte le stringe di risposta terminano con 0 byte. Se l'elemento di controllo ClientSocket riceve dati, esso richiama la corrispondente procedura.

```
procedure Twebio_ascii_client.ClientSocketThreadRun(Sender: TIdThreadComponent);
var
  Receivechar : string;
  SemiPosition : integer;
  OutputValue : word;
  InputValue : word;
begin
  Receivechar := ClientSocket.IOHandler.ReadString(1);
  if Receivechar[1] <> chr(0) then
    Receivestring := Receivestring + Receivechar
  else
  begin
    if Receivestring[1] = 'o' then
    begin
      OutputValue := StrtoInt('$'+copy(ReceiveString,
       pos(';',ReceiveString)+1,length(ReceiveString)-pos(';',ReceiveString)));
      if OutputValue and 1 = 1 then
        cb_output0.Checked := True
      else
        cb_output0.Checked := False;
      if OutputValue and 2 = 2 then
        cb_output1.Checked := True
      else
        cb_output1.Checked := False;
    end;
    if Receivestring[1] = 'i' then
    begin
      InputValue := StrtoInt('$'+copy(ReceiveString,
         pos(';',ReceiveString)+1,
         length(ReceiveString)-pos(';',ReceiveString)));
      if InputValue and 1 = 1 then
        cb_input0.Checked := True
      else
        cb_input0.Checked := False;
      if InputValue and 2 = 2 then
        cb_input1.Checked := True
      else
        cb_input1.Checked := False;
    end;
    if Receivestring[1] = 'c' then
    begin
      if copy(ReceiveString, 8, 1) = '0' then
        ed_counter0.Text := copy(ReceiveString, 10,length(ReceiveString)-9);
      if copy(ReceiveString, 8, 1) = '1' then
        ed_counter1.Text := copy(ReceiveString, 10,length(ReceiveString)-9);
      if copy(ReceiveString, 8, 1) = ';' then
      begin
        ReceiveString[8] := ' ';
        ed_counter0.Text := copy(ReceiveString, 9, pos(';',ReceiveString)-9);
        ed_counter1.Text := copy(ReceiveString,
          pos(';',ReceiveString)+1,
          length(ReceiveString)-pos(';',ReceiveString));
      end;
    end;
    receivestring := '';
  end;
end;
```

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. Poiché lo stato degli output viene fornito in formato esadecimale, è necessaria una trasformazione in un valore integro per calcolare passaggi ulteriori. Ciò avviene anteponendo '$' nella StrToInt funzione. Nei counter è possibile sia interrogare i valori dei singoli contatori sia leggere tutti i counter in una volta sola. Gli stati dei singoli contatori vengono visualizzati in una stringa in decimali separati mediante punto e virgola.

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.

```
procedure Twebio_ascii_client.timer_pollingTimer(Sender: TObject);
begin
  if ClientSocket.Connected and cb_output_polling.Checked then
    ClientSocket.IOHandler.Write('GET /output?PW=' + ed_password.Text + '&');
  if ClientSocket.Connected and cb_input_polling.Checked then
    ClientSocket.IOHandler.Write('GET /input?PW=' + ed_password.Text + '&');
  if ClientSocket.Connected and cb_counter_polling.Checked then
    ClientSocket.IOHandler.Write('GET /counter?PW=' + ed_password.Text + '&');
end;
```

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

```
procedure Twebio_ascii_client.ed_intervalChange(Sender: TObject);
begin
  timer_polling.Interval := strtoint(ed_interval.Text);
end;
```

Il programma esempio supporta tutte le comuni funzioni del Web-IO nella modalità stringa di comando, ottimizzata per il Web-IO digitale 4.0 2x input, 2x output . Gli altri modelli Web-IO devono eventualmente essere adattati 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.

Download esempio di programma

Prodotti

Web-IO 4.0 digitale
2x input, 2x output

All'occorrenza possibilità di alimentazione anche tramite PoE

Web-IO 4.0 digitale
12x input, 12x output

12x ingressi,
12x uscite

Altri web-IO

Tutti i Web-IO digitali W&T da 24 V

W&T
www.WuT.de

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