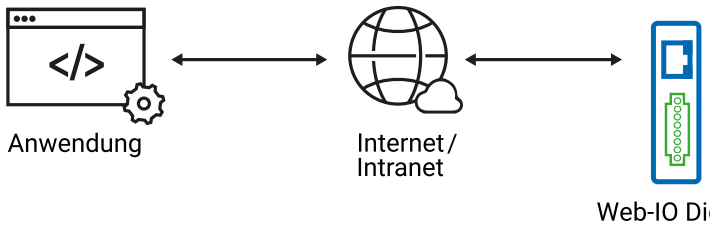
Tutorial zum Web-IO Digital:

Web-IO Digital mit Delphi steuern und überwachen

Produktübersicht
Applikationsübersicht

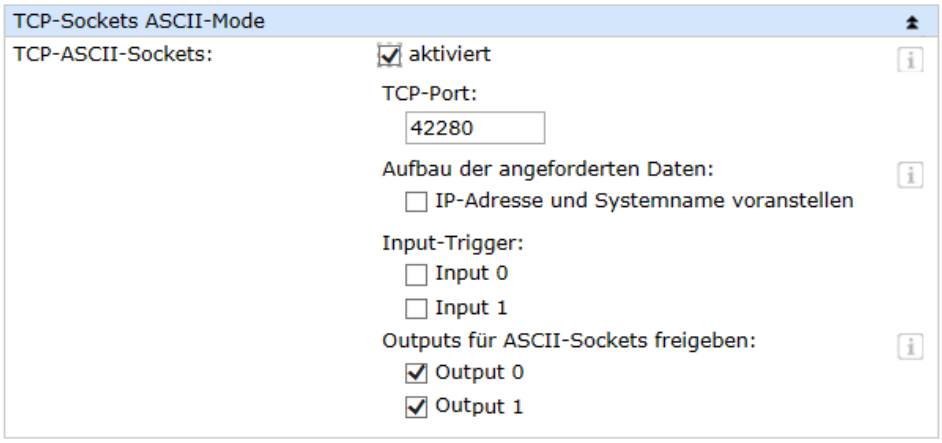
Als einfach zu erlernende Hochsprache bietet Delphi alles, was zum Programmieren von TCP/IP-Anwendungen nötig ist. Damit ist Delphi auch ein beliebtes Hilfsmittel, um Anwendungen zu erstellen, die mit dem Web-IO Digital kommunizieren. Zusätzliche Treiber oder DLLs werden nicht benötigt.



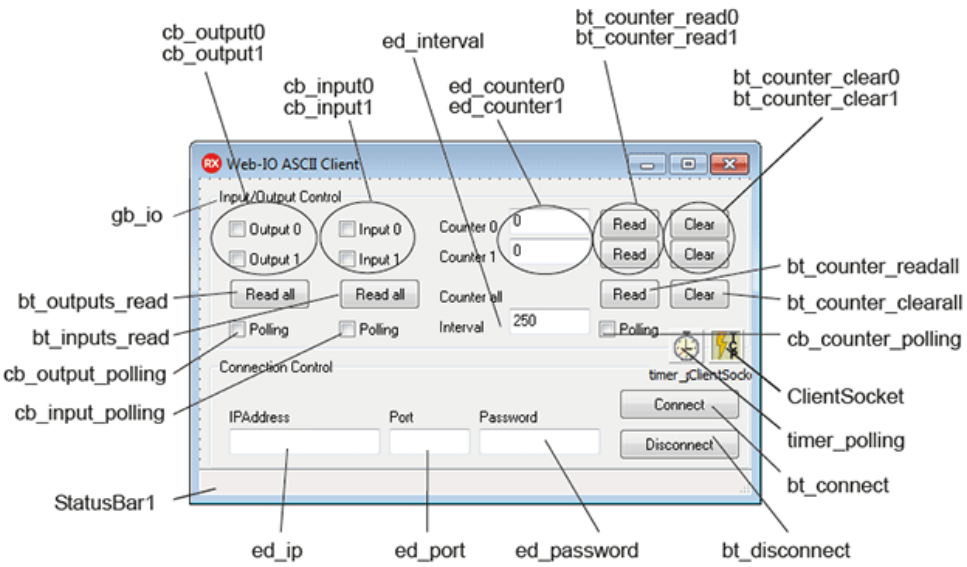
Mit dem folgenden Programmbeispiel können Sie Ihr Web-IO Digital mit seinen Inputs und Outputs in einer Windows-Anwendung abbilden. Darüber hinaus können Sie die Outputs des Web-IO schalten.

Vorbereitungen

- Web-IO mit Spannung versorgen und IOs verdrahten
- Web-IO mit dem Netzwerk verbinden
- IP-Adressen vergeben
- Beim Web-IO im Bereich *Kommunikationswege >> Socket-API* die *TCP-ASCII-Sockets* aktivieren und die *Outputs zum Schalten freigeben*



Zusammenstellen der verschiedenen Bedienelemente und Anzeigeobjekte im Delphi-Form



Bei der Benennung der einzelnen Objekte ist es hilfreich, sinngebende Namen zu verwenden. In diesem Beipiel beschreibt der erste Teil des Namens die Art des Objektes und der zweite Teil die Funktion.

Für die Abwicklung der TCP-Kommunikation wird die *TIdTCPClient*-Komponente von Indy genutzt, die Bestandteil der Delphi-Komponenten ist.

Globale Variablen und Prozeduren

Zunächst müssen einige globale Variablen deklariert werden.

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

Wichtig ist die Thread-Komponente *ClientSocketThread*, für die auch noch eine entsprechende Prozedur angelegt werden muss. Später wird dieser Thread den asynchronen Empfang eingehender Netzwerkdaten abwickeln.

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

Die entsprechende Prozedur muss im *implementation*-Bereich angelegt werden (siehe weiter unten).

Programmstart

Einrichten der Bedienelemente
Die Gruppe mit den Bedienelementen für das Web-IO wird zunächst für die Bedienung gesperrt. In der Statuszeile wird angezeigt, dass noch keine Verbindung besteht.

```
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;
```

Die Verbindungskontrolle

Einleiten der Verbindung
Durch Eingabe der IP-Adresse des Web-IO in das Textfeld *ed_ip* und des TCP-Ports in das Textfeld *ed_port* und Klick auf den Button *bt_connect* wird der Verbindungsaufbau gestartet.

```
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;
```

Verbindung kommt zustande
Sobald das Web-IO die Verbindung annimmt, führt das ClientSocket-Steuerelement die entsprechende Prozedur aus. In der Statuszeile wird das Zustandekommen der Verbindung angezeigt, die Bedienelemente werden zur Benutzung freigegeben und der Disconnect-Button wird bedienbar.

```
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;
```

Trennen der Verbindung
Die Verbindung bleibt solange bestehen, bis sie vom Benutzer durch Klick auf den Disconnect-Button beendet wird oder das Web-IO die Verbindung beendet.

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

Auch in diesem Fall ruft das ClientSocket-Steuerelement eine entsprechende Prozedur auf.

```
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;
```

Bedienung und Kommunikation von Client-Seite

Sobald eine Verbindung mit dem Web-IO zustande gekommen ist, kann der Anwender durch Bedienung der entsprechenden Programmelemente Kommandos an das Web-IO senden.

Setzen der Outputs
Das Setzen der Outputs wird dem Anwender über zwei Checkboxen *cb_outputx* ermöglicht. Das Programm nutzt dazu das MouseUP-Ereignis dieses Objektes. Wird ein Mouse Up, also ein Loslassen der Output-Checkbox registriert, führt das Programm die entsprechende Prozedur aus und gibt - je nachdem, ob die Checkbox gesetzt ist oder nicht - das passende Kommando an das Web-IO weiter.

```
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;;
```

Output/Input-Status abfragen
Den Status der Outputs und Inputs kann der Anwender durch Anklicken des zugehörigen Buttons anfordern.

```
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;
```

Counter abfragen löschen
Auch die Zählerstände der Input-Counter lassen sich abfragen bzw. löschen.

```
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;
```

Natürlich lassen sich auch alle Counter zeitgleich lesen bzw. löschen.

```
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;
```

Datenempfang vom Web-IO

Auswerten und Anzeigen der empfangenen Daten
Alle Kommandos und Anfragen an das Web-IO werden mit einem Antwort-String quittiert. Für den Empfang und die Auswertung der Antworten ist der anfangs erwähnte Thread *ClientSocketThreadRun* verantwortlich.

Format der Antworten
Die Antworten haben je nach Type einen spezifischen Aufbau.

- Für die Outputs: `output;<Binärwert des Outputstatus im hexadezimalen Format>`
- Für die Inputs: `input;<Binärwert des Outputstatus im hexadezimalen Format>`
- Für die Counter: `counterx;<dezimaler Zählerstand>`
- oder `counter;<dezimaler Zählerstand 0 >; <dezimaler Zählerstand 0 >; .....`,
  wenn alle Counter auf einmal gelesen werden sollen.

Alle Antwort-Strings sind mit einem 0-Byte abgeschlossen. Werden vom ClientSocket-Steuerelement Daten empfangen, ruft dieses die entsprechende Prozedur auf.

```
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;
```

Die Empfangsprozedur überprüft anhand des ersten Zeichens der Empfangsdaten, ob es um Input-, Output- oder Counter-Meldungen geht. Abhängig davon wird z. B. festgestellt, welcher Output welchen Status hat. Da der Output-Status hexadezimal übergeben wird, ist eine Wandlung in einen Integerwert nötig, um weitere Schritte zu berechnen. Das erfolgt über das Voranstellen von '$' bei der StrToInt-Funktion. Bei den Countern ist es sowohl möglich, einzelne Zählerwerte abzufragen, als auch alle Counter in einem Zug auszulesen. Die einzelnen Zählerstände werden dann dezimal mit Semikolon getrennt in einem String ausgegeben.

Polling

Zyklisches Abfragen bestimmter Werte
Um auch eine automatische Aktualisierung der Anzeige zu ermöglichen, wird ein Timer benutzt.

In Abhängigkeit der Checkboxen für Output-, Input- und Counter-Polling werden die entsprechenden Informationen im eingestellten Intervall vom Web-IO abgerufen.

```
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;
```

Das gewünschte Intervall kann in das entsprechende Textfeld eingegeben werden. Bei Änderung wird das Timer-Intervall dann automatisch angepasst.

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

Das Beispielprogramm unterstützt alle gängigen Funktionen des Web-IO im Kommando-String-Modus, optimiert für das Web-IO Digital 4.0 2xIn, 2xOut. Für die anderen Web-IO-Modelle müssen ggf. Anpassung am Programm vorgenommen werden. Weitere Programmbeispiele zur Socket-Programmierung finden Sie auf den Tool-Seiten zum Web-IO. Eine detaillierte Beschreibung zur Socket-Schnittstelle der Web-IO Digital-Modelle finden Sie im Referenzhandbuch.

Programmbeispiel herunterladen

Produkte

Web-IO 4.0 Digital 2xIn, 2xOut

Bei Bedarf auch über PoE zu versorgen

Web-IO 4.0 Digital 12xIn, 12xOut

12x Eingänge, 12x Ausgänge

Weitere Web-IOs

Alle W&T Web-IO Digital 24V

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis
GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17 Uhr)
Fax: 0202/2680-265
info@wut.de